

Gary M. Hoffman (*Pro Hac Vice*)
Kenneth W. Brothers (*Pro Hac Vice*)
DICKSTEIN SHAPIRO LLP
1825 Eye Street, NW
Washington, DC 20006-5403
Phone (202) 420-2200
Fax (202) 420-2201

Edward A. Meilman (*Pro Hac Vice*)
DICKSTEIN SHAPIRO LLP
1177 Avenue of the Americas
New York, New York 10036-2714
Phone (212) 277-6500
Fax (212) 277-6501

Jeffrey B. Demain, State Bar No. 126715
Jonathan Weissglass, State Bar No. 185008
ALTSHULER, BERZON, NUSSBAUM, RUBIN & DEMAIN
177 Post Street, Suite 300
San Francisco, California 94108
Phone (415) 421-7151
Fax (415) 362-8064

Attorneys for Ricoh Company, Ltd.

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,)
Plaintiff,)
vs.)
AEROFLEX ET AL.,) CASE NO. CV 03-4669 MJJ (EMC)
Defendants.) CASE NO. CV 03-2289 MJJ (EMC)

SYNOPSYS, INC.,) **DECLARATION OF DEANNA ALLEN IN
Plaintiff,) SUPPORT OF RICOH'S OPPOSITION TO
vs.) DEFENDANTS' MOTION FOR RULE 11
RICOH COMPANY, LTD.,) SANCTIONS
Defendant.)**

1 DeAnna Allen declares as follows:

2 1. My name is DeAnna Allen, an attorney with the law firm of Dickstein Shapiro LLP,
3 counsel for Ricoh Company Limited. I am over the age of 21 and am competent to make this
4 declaration. This declaration is submitted in support of Ricoh's Opposition to Defendants' Motion for
5 Rule 11 Sanctions. Based on my personal knowledge and information except as stated upon information
6 and belief, I hereby declare to all the facts in this declaration.

7 2. Attached as Exhibit 1 is a true and correct copy of the Subpoena served on the University
8 of Michigan by Defendants on August 14, 2006.

9 3. Attached as Exhibit 2 is a true and correct copy of the Subpoena served on Yale
10 University by Defendants on August 14, 2006..

11 4. Attached as Exhibit 3 is a true and correct copy of the Subpoena served on the
12 Massachusetts Institute of Technology by Defendants on August 14, 2006..

13 5. Exhibit 4 is a true and correct copy of the transcript of the September 12, 2005, meet and
14 confer (Tapes 1-4) between counsel for Ricoh and Defendants.

15 6. Exhibit 5 is a true and correct copy of the transcript of the September 14, 2005, meet and
16 confer (Tapes 5-6) between counsel for Ricoh and Defendants.

17 7. Exhibit 6 is a true and correct copy of the Transcript of the Deposition of Russell Segal,
18 dated December 20, 2005, who was deposed by Ricoh as a Rule 30(b)(6) witness on behalf of Synopsys.

19 8. Attached as Exhibit 7 is a true and correct copy of U.S. Patent No. 6,226,776 issued May
20 1, 2001.

21 9. Exhibit 8 is a true and correct copy of the Transcript of the Deposition of Robert Smith,
22 dated February 9, 2006, who was deposed by Ricoh as a Rule 30(b)(6) witness on behalf of AMI
23 Semiconductor.

24 10. Exhibit 9 is a true and correct copy of the Transcript of the Deposition of Robert Smith,
25 dated February 10, 2006, who was deposed by Ricoh as a Rule 30(b)(6) witness on behalf of AMI
26 Semiconductor.

1 Signed at Washington, D.C. on September 26, 2006.
2
3 September 26, 2006 /s/ DeAnna Allen
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

DeAnna Allen

1 Teresa M. Corbin (SBN 132360)
2 Denise M. De Mory (SBN 168076)
3 Ethan B. Andelman (SBN 209101)
4 Jaclyn C. Fink (SBN 217913)
5 HOWREY LLP
 525 Market Street, Suite 3600
 San Francisco, California 94105
 Telephone: (415) 848-4900
 Facsimile: (415) 848-4999

6 Attorneys for Plaintiff SYNOPSYS, INC.
7 and for Defendants AEROFLEX INCORPORATED,
8 AMI SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS, LTD., MATROX
GRAPHICS, INC., MATROX INTERNATIONAL
CORP., MATROX TECH, INC., and
9 AEROFLEX COLORADO SPRINGS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.

Case No. C03-04669 MJJ (EMC)

Plaintiff.

Case No. C03-02289 MJJ (EMC)

VS.

16 AEROFLEX INCORPORATED, AMI
SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS LTD., MATROX
17 GRAPHICS INC., MATROX INTERNATIONAL
CORP., MATROX TECH, INC., AND
18 AEROFLEX COLORADO SPRINGS, INC.

**NOTICE OF SUBPOENA TO THE
UNIVERSITY OF MICHIGAN**

Defendants.

SYNOPSYS, INC.

Plaintiff,

VS.

RICOH COMPANY, LTD.,

Defendant.

1 TO ALL PARTIES AND THEIR ATTORNEYS OF RECORD:

2 YOU ARE HEREBY NOTIFIED that pursuant to Federal Rule of Civil Procedure 45, Plaintiff

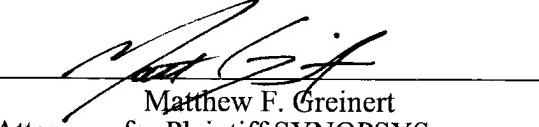
3 Synopsys, Inc. has served the University of Michigan c/o Donica Thomas Varner, Assistant General

4 Counsel, the attached subpoena for production of documents.

5 The University is required to produce documents in its custody, possession or control specified in
6 Attachment A to the subpoena by 10:00 a.m. CST on Friday, August 18, 2006 at American Reprographics
7 Systems, 660 Woodward Avenue, Suite 610, Detroit, Michigan, 48226, Telephone: (313) 965-5090.

8 Dated: August 14, 2006

HOWREY LLP

9
10 By: 

Matthew F. Greinert

11 Attorneys for Plaintiff SYNOPSYS
12 Defendants AEROFLEX
13 INCORPORATED, AMI
14 SEMICONDUCTOR, INC., MATROX
15 ELECTRONIC SYSTEMS, LTD.,
16 MATROX GRAPHICS, INC., MATROX
17 INTERNATIONAL CORP., MATROX
18 TECH, INC., and AEROFLEX
19 COLORADO SPRINGS, INC.

PROOF OF SERVICE

2 STATE OF CALIFORNIA)
 3 COUNTY OF SAN FRANCISCO) ss.:
 4

5 I am employed in the County of San Francisco, State of California. I am over the age of 18 and not
 a party to the within action. My business address is 525 Market Street, Suite 3600, San Francisco,
 California 94105.

6 On August 14, 2006 I served on the interested parties in said action the within:

NOTICE OF SUBPOENA TO THE UNIVERSITY OF MICHIGAN

8 by causing said document to be sent by Electronic Mail on August 14, 2006 to the email addresses
 indicated for the parties listed below and by placing a true copy thereof in a sealed envelope(s) addressed as
 9 stated below and causing such envelope(s) to be delivered as follows:

10 Gary M. Hoffman, Esq.
HoffmanG@dsmo.com
 11 Dickstein Shapiro Morin & Oshinsky, LLP
 2101 L Street, N.W.
 Washington, DC 20037-1526

Jeffrey Demain, Esq.
jdemain@altshulerberzon.com
 Altshuler, Berzon, Nussbaum, Rubin & Demain
 177 Post Street, Suite 300
 San Francisco, CA 94108

13 Facsimile No.: (202) 887-0689

Facsimile No.: (415) 362-8064

14 Edward A. Meilman, Esq.
MeilmanE@dsmo.com
 15 Dickstein Shapiro Morin & Oshinsky, LLP
 1177 Avenue of the Americas
 New York, NY 10036-2714

16 Facsimile No.: (212) 896-5471

17 (OVERNIGHT DELIVERY) on August 14, 2006 by depositing in a box or other facility regularly
 maintained by Federal Express, an express service carrier, or delivering to a courier or driver
 authorized by said express service carrier to receive documents, a true copy of the foregoing
 document in sealed envelopes or packages designated by the express service carrier, addressed as
 stated above, with fees for overnight delivery paid or provided for and causing such envelope(s) to
 be delivered by said express service carrier on.

21 I declare under penalty of perjury that I am employed in the office of a member of the bar of this
 Court at whose direction the service was made and that the foregoing is true and correct.

22 Executed on August 14, 2006, at San Francisco, California.

24 _____
 James M. James
 (Type or print name)


 25 _____
 (Signature)

Issued by the

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF MICHIGAN

RICOH COMPANY, LTD

V.
 AEROFLEX, INCORPORATED, et al.

SUBPOENA IN A CIVIL CASE

Case Number: ¹ C03-04669 MJJ (EMC)
 (Pending in the US District Court for
 the Northern District of California)

TO: DONICA THOMAS VARNER, Esq., Assistant General Counsel,
 on behalf of UNIVERSITY OF MICHIGAN
 4010 Fleming Building
 503 Thompson Street, Ann Arbor, MI 48109

YOU ARE COMMANDED to appear in the United States District court at the place, date, and time specified below to testify in the above case.

PLACE OF TESTIMONY	COURTROOM
	DATE AND TIME

YOU ARE COMMANDED to appear at the place, date, and time specified below to testify at the taking of a deposition in the above case.

PLACE OF DEPOSITION	DATE AND TIME

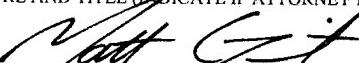
YOU ARE COMMANDED to produce and permit inspection and copying of the following documents or objects at the place, date, and time specified below (list documents or objects):

SEE ATTACHMENT A

PLACE American Reprographics Systems, attn: Jim Higgins 660 Woodward Avenue, Suite 610, Detroit, Michigan 48226, (313) 965-5090	DATE AND TIME August 18, 2006, 10:00 am CST
<input type="checkbox"/> YOU ARE COMMANDED to permit inspection of the following premises at the date and time specified below.	

PREMISES	DATE AND TIME

Any organization not a party to this suit that is subpoenaed for the taking of a deposition shall designate one or more officers, directors, or managing agents, or other persons who consent to testify on its behalf, and may set forth, for each person designated, the matters on which the person will testify. Federal Rules of Civil Procedure, 30(b)(6).

ISSUING OFFICER'S SIGNATURE AND TITLE (INDICATE IF ATTORNEY FOR PLAINTIFF OR DEFENDANT) Attorney for Defendant 	DATE August 14, 2006
--	-------------------------

ISSUING OFFICER'S NAME, ADDRESS AND PHONE NUMBER
 Matthew F. Greinert, HOWREY LLP, 525 Market Street, Suite 3600, San Francisco, CA 94105; Telephone: (415) 848-4900

(See Rule 45, Federal Rules of Civil Procedure, Parts C & D on next page)

¹ If action is pending in district other than district of issuance, state district under case number.

PROOF OF SERVICE

DATE	PLACE
------	-------

SERVED:

SERVED ON (PRINT NAME)	MANNER OF SERVICE
------------------------	-------------------

SERVED BY (PRINT NAME)	TITLE
------------------------	-------

DECLARATION OF SERVER

I declare under penalty of perjury under the laws of the United States of America that the foregoing information contained in the Proof of Service is true and correct.

Executed on _____

DATE

SIGNATURE OF SERVER

ADDRESS OF SERVER

Rule 45, Federal Rules of Civil Procedure, Parts C & D:**(c) PROTECTION OF PERSONS SUBJECT TO SUBPOENAS.**

(1) A party or an attorney responsible for the issuance and service of a subpoena shall take reasonable steps to avoid imposing undue burden or expense on a person subject to that subpoena. The court on behalf of which the subpoena was issued shall enforce this duty and impose upon the party or attorney in breach of this duty an appropriate sanction which may include, but is not limited to, lost earnings and reasonable attorney's fee.

(2) (A) A person commanded to produce and permit inspection and copying of designated books, papers, documents or tangible things, or inspection of premises need not appear in person at the place of production or inspection unless commanded to appear for deposition, hearing or trial.

(B) Subject to paragraph (d) (2) of this rule, a person commanded to produce and permit inspection and copying may, within 14 days after service of subpoena or before the time specified for compliance if such time is less than 14 days after service, serve upon the party or attorney designated in the subpoena written objection to inspection or copying of any or all of the designated materials or of the premises. If objection is made, the party serving the subpoena shall not be entitled to inspect and copy materials or inspect the premises except pursuant to an order of the court by which the subpoena was issued. If objection has been made, the party serving the subpoena may, upon notice to the person commanded to produce, move at any time for an order to compel the production. Such an order to comply production shall protect any person who is not a party or an officer of a party from significant expense resulting from the inspection and copying commanded.

(3) (A) On timely motion, the court by which a subpoena was issued shall quash or modify the subpoena if it

(i) fails to allow reasonable time for compliance,

(ii) requires a person who is not a party or an officer of a party to travel to a place more than 100 miles from the place where that person resides, is employed or regularly transacts business in person, except that, subject to the provisions of clause (c) (3) (B) (iii) of this rule, such a person may in order to attend

trial be commanded to travel from any such place within the state in which the trial is held, or

(iii) requires disclosure of privileged or other protected matter and no exception or waiver applies, or

(iv) subjects a person to undue burden.

(B) If a subpoena

(i) requires disclosure of a trade secret or other confidential research, development, or commercial information, or

(ii) requires disclosure of an unretained expert's opinion or information not describing specific events or occurrences in dispute and resulting from the expert's study made not at the request of any party, or

(iii) requires a person who is not a party or an officer of a party to incur substantial expense to travel more than 100 miles to attend trial, the court may, to protect a person subject to or affected by the subpoena, quash or modify the subpoena, or, if the party in whose behalf the subpoena is issued shows a substantial need for the testimony or material that cannot be otherwise met without undue hardship and assures that the person to whom the subpoena is addressed will be reasonably compensated, the court may order appearance or production only upon specified conditions.

(d) DUTIES IN RESPONDING TO SUBPOENA.

(1) A person responding to a subpoena to produce documents shall produce them as they are kept in the usual course of business or shall organize and label them to correspond with the categories in the demand.

(2) When information subject to a subpoena is withheld on a claim that it is privileged or subject to protection as trial preparation materials, the claim shall be made expressly and shall be supported by a description of the nature of the documents, communications, or things not produced that is sufficient to enable the demanding party to contest the claim.

ATTACHMENT A

Pursuant to Federal Rule of Civil Procedure 45 and as directed in the subpoena attached hereto, you are to produce all documents and things within the scope of the following definitions and descriptions that are within your possession, custody, or control. A Protective Order has been entered in this case by the United States District Court for the Northern District of California and is attached as Attachment B. Included in the Protective Order are provisions for the protection of confidential information produced by a third party. With respect to documents and things withheld under a claim of privilege, you are required under Rule 45 to describe the nature of the documents and things withheld in a manner sufficient to enable the demanding party to contest the claims.

DEFINITIONS

1. The terms "you," and "your," mean, without limitation, the University of Michigan, including without limitation all of its subsidiaries, parents, departments and affiliates, and all past or present directors, officers, agents, representatives, employees, students, consultants, attorneys, entities acting in joint-venture or partnership relationships with the University of Michigan and others acting on behalf of the University of Michigan.

2. As used herein, the word "document" means the original and each non-identical copy of any written, printed, typed, recorded, computerized, electronic, taped, graphic, or other matter, in whatever form, whether in final or draft, including but not limited to all materials that constitute "writings," "recordings," "photographs," "source code" or "executable code" within the broadest meaning of Rule 1001 of the Federal Rules of Evidence and all materials that constitute "documents" within the broadest meaning of Rule 34 of the Federal Rules of Civil Procedure. The word "document" includes, without limitation, printed matter, electronic mail, materials stored on computer hard drives, diskettes, tapes, any other computer media, and any other information stored magnetically, optically or in any electronic medium and/or form.

3. As used herein, "person" means any individual, firm, partnership, corporation, proprietorship, association, governmental body, or any other organization or entity.

4. As used herein, "communication" includes, without limitation, communications by whatever means transmitted (i.e., whether oral, written, electronic or other methods used), as well as any note, memorandum or other record thereof.

5. The terms "regarding, referring or relating to" and "concerning" mean reflecting, concerning, containing, pertaining, referring, relating to, indicating, showing, describing, evidencing, discussing, mentioning, embodying or computing.

6. Whenever the singular is used, it shall also be taken to include the plural, and vice versa. Whenever the conjunctive is used, it shall also be taken to include the disjunctive, and vice versa.

INSTRUCTIONS

The following instructions apply to each of the requests for documents set forth herein:

1. Please produce entire documents, including, but not limited to, attachments, enclosures, cover letters, memoranda, and appendices.

2. Pursuant to Rule 26(e) of the Federal Rules of Civil Procedure, these requests for documents shall be deemed continuous up to and following the trial of this proceeding such that any documents or things requested herein which is either discovered by you or comes within your possession, custody or control subsequent to your initial responses hereto but prior to the final conclusion of this case should be produced in a supplemental response to these Document Requests immediately upon its discovery or receipt by you or your counsel.

3. If any document is withheld under a claim of privilege, in order that the Court and the parties may determine the validity of the claim of privilege, please provide a privilege log identifying each document withheld, including

- a. The type of document;
- b. The approximate date, and manner of recording, creating or otherwise preparing the document;
- c. The subject matter of the document;

- d. The name and organizational position of the person(s) who produced the document,
- e. The name and organizational position of the person(s) who received a copy of the document, or to whom the document was disclosed; and
- f. The claimed grounds on which the document is being withheld and facts sufficient to show the basis for each claim of privilege.

4. If you object to any part of a request for documents and refuse to produce documents responsive to that part, state your objection and respond to the remaining portion of that request. If you object to the scope or time period of a request for documents, state your objection and respond to the request for documents for the scope or time period you believe is appropriate.

5. Please produce all documents in the order in which they are kept in the ordinary course of business, and in their original file folders, binders, covers or containers, or facsimile thereof.

6. Any document bearing any changes, including, but not limited to, markings, handwritten notation, or other differences, that are not a part of the original text, or any reproduction thereof, is to be considered a separate document for purposes of responding to the following document requests. English translations of partial translations of foreign language documents should also be considered separate documents.

7. If a requested document is in a language other than English, please produce both the original and any existing English translation thereof.

8. If any of the following requests for documents cannot be responded to in full after exercising due diligence to secure the requested documents, please so state and respond to the extent possible, specifying your inability to respond to the remainder and stating whatever information you have regarding, referring or relating to the unanswered portions. If your response is qualified in any particular manner, set forth the details of such qualification.

9. Please produce hard copies of electronic records or produce computerized information in an intelligible format with a description of the system from which it was derived sufficient to permit rendering the materials intelligible.

DOCUMENT REQUESTS

Request No.1:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Introduction to Computer Organization" (EECS 370) in the Electrical Engineering and Computer Science Department at the University of Michigan, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 2:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Logic Circuit Synthesis and Optimization" (EECS 478) in the Electrical Engineering and Computer Science Department at the University of Michigan, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 3:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Computer-Aided Design of Embedded Systems" in the Electrical Engineering and Computer Science Department at the University of Michigan, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 4:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of any course within the Electrical Engineering and Computer Science Department, or any other Department, at the University of Michigan, including but not limited to, course syllabi, handouts,

outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for any such course.

Request No. 5:

All documents regarding, referring or relating to any invention disclosures submitted to the University of Michigan Technology Transfer Office by Dr. Marios Papaefthymiou while being employed by the University of Michigan, including but not limited to, Invention Disclosure Nos. 1496, 1759, 2270, 2299, 2300, and 2452.

ATTACHMENT B

1
2
3
4
5
6
7
8 UNITED STATES DISTRICT COURT
9 NORTHERN DISTRICT OF CALIFORNIA
10 SAN FRANCISCO DIVISION

11
12
13 SYNOPSY, INC., }
14 Plaintiff }
15 vs. }
16 RICOH COMPANY, LTD., }
17 Defendant. }

CASE NO. C03-02289-MJJ (EMC)
STIPULATED PROTECTIVE ORDER

18
19 1. All Confidential Information produced or exchanged in the course of this litigation shall
be used solely for the purpose of preparation and trial of this litigation and for no other purpose
whatsoever, and shall not be disclosed to any person except in accordance with the terms hereof.

20
21 2. "Confidential Information," as used herein, means any information of any type, kind or
character that is designated as "Confidential" by any of the supplying or receiving parties, whether it be
a document, information contained in a document, information revealed during a deposition,
information revealed in an interrogatory answer or otherwise. In designating information as
"Confidential," a party will make such designation only as to that information that it in good faith
believes contains "Confidential Information."

22
23 3. (a) "Confidential Information" includes, but is not limited to, (i) proprietary technical

1
2 information and specifications, (ii) trade secrets (iii) confidential know-how, and (iv) proprietary
3 business and financial information and any other non-public information, the disclosure of which is
4 likely to have the effect of causing significant competitive harm to the disclosing party or party from
5 which the information was obtained. Nothing in this paragraph shall be construed to limit the
description of "Confidential Information" set forth in paragraph 2.

8 (b) Nothing shall be regarded as "Confidential Information" if it is information that:
9 (i) is in the public domain at the time of disclosure, as evidenced by a written document;
10 (ii) becomes part of the public domain through no fault of the other party, as evidenced
11 by a written document;
12 (iii) was in the receiving party's rightful and lawful possession at the time of disclosure,
13 as evidenced by a written document; or
14 (iv) is lawfully received by the receiving party from a third party at a later date without
15 restriction as to disclosure, provided such third party has the right to make the disclosure to the
16 receiving party.
17

18 4. "Qualified Persons," as used herein means:
19 (a) To the Court and its officers and staff, including court reporters;
20 (b) Outside attorneys of record for the parties in this litigation and employees of such
attorneys to whom it is necessary that the material be shown for purposes of this litigation;
22 (c) Outside experts, consultants, advisors or investigators (collectively referred to
hereafter²³ as "experts") who have signed an undertaking pursuant to paragraph 5 but only after
compliance with the provisions of paragraph 5 below;
25 (d) To non-party support services including, but not limited to, court reporters, outside
copy services, document imaging and database services, design services who have signed confidentiality
agreements, jury consultants who have signed confidentiality agreements, mock jurors who have signed
confidentiality agreements, and language translators who have signed confidentiality agreements
28

1
 2 (including support staff) as may be reasonably necessary in connection with the preparation or conduct
 3 of this action;
 4

5 (e) Anyone to whom the parties consent in writing; and
 6 (f) If this Court so elects, any other person may be designated as a Qualified Person by
 order of this Court, after notice and opportunity to be heard to all parties.

8 5. Prior to the disclosure of any "Confidential Information" to any expert under Paragraph
 4(c), counsel for the Party seeking to make the disclosure shall: (i) deliver a copy of this Protective
 Order as entered to such person, explain its terms to such person, and secure the signature of such
 10 person on a written undertaking in the form attached hereto as Exhibit A, and (ii) transmit by facsimile
 11 and mail to counsel for the other Parties a copy of the signed Exhibit A, accompanied by a curriculum
 12 vitae, at least ten (10) calendar days before any "Confidential Information" designated under this
 13 Protective Order is to be disclosed to the signator. The curriculum vitae should identify the general
 14 area(s) of expertise of the expert, provide a brief job history, specify all employment, expert or
 15 consulting engagements by the expert within the past five (5) years, and state all present or prior
 16 relationships between the expert and any entity directly or indirectly involved in this litigation or
 17 providing an indemnity to any such entity, its subsidiaries or its affiliates. Any Party may object to the
 18 proposed disclosure to an expert within the ten (10) calendar day period following the transmittal of
 19 Exhibit A and the curriculum vitae, by stating specifically in writing the reasons why the Party believes
 20 such expert should not receive designated "Confidential Information." If during that ten (10) calendar
 21 day period a Party makes such a written objection, there shall be no disclosure of "Confidential
 22 Information" to the expert absent mutual agreement of the Parties, waiver of the objection as stated
 23 below, or further order of the Court. After a Party objects to the proposed disclosure to an expert, the
 24 objecting Party shall move, by noticed motion or by ex parte application, for an order that disclosure not
 25 be made to such expert within five (5) business days following the date that the objection is made, or the
 26
 27 Party's objection shall be deemed waived and disclosure may be made to the expert. The burden shall
 28

1

2

be on the objecting Party to establish why the disclosure should not be made. Each Party shall maintain
3

a file of all such signed copies of Exhibit A. However, it shall not be necessary for administrative,
4

secretarial or clerical personnel working for such Qualified Person to sign a written undertaking.

6 6. (a) Documents produced in this action may be designated by any party or parties as
“Confidential” by marking each page of the document(s) with the designation “Confidential.”

8 (b) In lieu of marking the original of a document, if the original is not produced, the
designating party may mark the copies that are produced or exchanged. Originals shall be preserved for
inspection.¹⁰

11 (c) If the document is not in paper form, the producing person or entity shall use other
12 such reasonable means as necessary to identify clearly the document or information as “Confidential.”

13 7. Discovery responses or other litigation materials may be designated by any party or
14 parties as “Confidential” by marking each page of the response with the designation “Confidential.”

15 8. The designation of information disclosed during a deposition as “Confidential” shall be
made either by a statement on the record at the deposition or within twenty (20) calendar days after
17 receipt by counsel of a copy of the deposition transcript. Such designation will be applied to only those
portions of the deposition transcript that include a specific question and response or series of questions
and responses containing “Confidential Information.” The deposition transcript shall be printed in
consecutive pages (whether or not some pages are designated as “Confidential”) with a marking on the
cover of the deposition transcript indicating the “Confidential” designation contained therein. Unless
previously designated otherwise, all deposition transcripts shall be treated as “Confidential” in their
entirety prior to the end of the twenty (20) calendar day period following receipt by counsel of a copy of
the deposition transcript.²⁵

26 9. “Confidential Information” shall not be disclosed or made available by the receiving
27 party to persons other than Qualified Persons except that nothing herein is intended to prevent
28 individuals who are in-house counsel or a member of the professional legal department of the Parties

1

2

from having access to pleadings, briefs and exhibits or declarations filed with the Court and expert
3
reports, including exhibits that are designated as "Confidential."
4

5 10. (a) Documents to be inspected shall be treated as "Confidential" although such documents need not be marked as "Confidential" prior to inspection. At the time of copying for the receiving parties, any documents containing "Confidential Information" shall be stamped prominently "Confidential" by the producing party.

9 (b) Nothing herein shall prevent disclosure beyond the terms of this Order if each party
10 designating the information as "Confidential" consents to such disclosure or if the Court, after notice to all effected parties, orders such disclosures. Nothing herein shall prevent any counsel of record from
11 utilizing "Confidential Information" in the examination or cross-examination of any person who is
12 indicated on the document as being an author, source or recipient of the "Confidential Information,"
13 irrespective of which party produced such information. Nothing herein shall prevent any counsel of
14 record from utilizing "Confidential Information" in the examination or cross-examination of any person
15 who is a current or former officer, director or employee of the party so designating the information as
16 "Confidential" or of the party that produced the information or of a related entity.
17
18

19 11. If a party inadvertently discloses any document or thing containing information that it deems confidential without designating it as "Confidential," the disclosing party shall promptly upon discovery of such inadvertent disclosure inform the receiving party in writing, and the receiving party and all Qualified Persons possessing such information shall thereafter treat the information as
20 "Confidential" under this Order. To the extent such information may have been disclosed to persons other than Qualified Persons described in this document, the receiving party shall make every
21 reasonable effort to retrieve the information promptly from such persons and to avoid any further
22 disclosure to and by such persons.
23
24
25
26
27

12. A party shall not be obligated to challenge the propriety of a designation as
28 "Confidential" at the time made, and a failure to do so shall not preclude a subsequent challenge thereto.

1

2

Nor will the failure to object be construed as an admission that any particular "Confidential
 3
 Information" contains or reflects currently valuable trade secrets or confidential commercial
 4
 information. In the event that any party to this litigation disagrees at any stage of these proceedings
 with the designation by the designating party of any information as "Confidential," or the designation of
 any person as a Qualified Person, the parties shall first try to resolve such dispute in good faith on an
 informa&Basis, such as production of redacted copies. If the parties are unsuccessful in informally
 resolving any disputes regarding the designation of any document or information as "Confidential," the
 Court shall resolve all such disputes. It shall be the burden of the party making any designation to
 establish that the information so designated is "Confidential" within the meaning of this Protective
 Order.¹⁰ The "Confidential Information" that is the subject of the dispute shall be treated as originally
 11
 12 designated pending resolution of the dispute.

13

¹⁴ 13. The parties may, by written stipulation filed and approved by the Court, amend this
 15
 Order, and any party may seek an order of this Court modifying this Protective Order. The parties agree
 16
 to meet and confer prior to seeking to modify this Protective Order. In addition, the Court may modify
 17
 this Protective Order in the interest of justice or otherwise at the Court's discretion.
 18

¹⁹ 14. In the event a party wishes to use any "Confidential Information" in any affidavits,
 briefs, memoranda of law, or other papers filed with the Court in this litigation, such "Confidential
 20
 Information" used therein shall be filed under seal with the Court. In addition to placing documents in a
 sealed envelope with instructions that the document is filed pursuant to the Stipulated Protective Order
 and that²¹ the envelope is not to be opened absent further order of the Court, the envelope should be
 22
 labeled to identify the title of the case, the case number, and the title of the document.
 23

²⁴ 15. The Clerk of this Court is directed to maintain under seal all documents and transcripts of
 25 deposition testimony and answers to interrogatories, admissions and other pleadings filed under seal
 26
 27 with the Court in this litigation that have been designated, in whole or in part, as "Confidential" by a
 28
 party to this action.

1
2 16. If a Party intends to offer into evidence or otherwise disclose in open court any
3 "Confidential Information" designated by another person or entity, counsel for such Party shall notify
4 the designating person or entity that the Party intends to disclose "Confidential Information" in open
5 court prior to the disclosure, so that the designating person or entity may confer with the Court
6 concerning appropriate procedures for protecting its "Confidential Information."

8 17. In the event any person or party that has possession, custody, or control of any
9 information designated as "Confidential" pursuant to the terms of this Protective Order receives a
10 subpoena or other process or order to produce such information, such person or party shall notify by
11 mail within five (5) business days of the Party's receipt of the request, the counsel for the party or
12 persons claiming confidential treatment of the documents sought by such subpoenas or other process or
13 order, shall furnish such counsel with a copy of said subpoena or other process or order, and shall
14 cooperate with respect to any procedure sought to be pursued by the party whose interests may be
15 affected. The party asserting the "Confidential" treatment shall have the burden of defending against
16 such subpoena, process or order. The person or party receiving the subpoena or process or order shall
17 be entitled to comply with it except: (a) to the extent the party asserting the "Confidential" treatment is
18 successful in obtaining an order modifying or quashing it; and (b) in complying with the process or
19 order shall, at a minimum, seek to obtain "Confidential" treatment of the "Confidential Information"
20 before producing it in the other proceeding or action.

22 18. If the discovery process calls for the production of information that a Party or Non-Party
23 does not wish to produce because the Party or Non-Party believes its disclosure would breach an
24 agreement with another person or entity to maintain such information in confidence, the disclosing Party
25 or Non-Party promptly shall give written notice to the other person or entity that its information is
26 subject to discovery in this litigation, and shall provide such person or entity with a copy of this
27 Protective Order. When such written notice is given to the person or entity, the disclosing Party or Non-
28 Party will advise the potential receiving Party that such notice has been given. The person or entity

1
 2
 whose information may be subject to discovery shall have ten (10) business days from receipt of the
 3
 written notice in which to seek relief from the Court, if the person or entity so desires. If the ten (10)
 4
 business days elapse without the person or entity seeking relief from the Court, the requested
 5
 information shall be produced in accordance with the terms of this Protective Order.

7 19. In the event that additional persons or entities become Parties, none of such Parties'
 8 counsel, experts or consultants retained to assist said counsel, shall have access to "Confidential
 9 Information" produced by or obtained from any other producing person or entity until said Party has
 10 executed and filed with the Court its agreement to be fully bound by this Protective Order.

11 20. This Protective Order shall apply to the parties and any non-party from whom discovery
 12 may be sought and who desires protection for the discovery sought. Thus, any non-party requested or
 13 required to produce or disclose information in this proceeding, through subpoena or otherwise, may
 14 designate such information pursuant to the terms of this Protective Order.
 15

16 21. (a) Nothing herein requires disclosure of information, documents or things which the
 17 disclosing entity contends is protected from disclosure by the attorney-client privilege or the work-
 18 product exception. Nothing herein shall preclude any party from moving this Court for an order
 19 directing the disclosure of such information, documents or things.

20 (b) In the event that any privileged attorney-client or work product documents or things
 21 are inadvertently produced for inspection and/or provided, the disclosing party shall identify such
 22 documents or things within five (5) days of when it discovers that the privileged materials were
 23 inadvertently produced for inspection and/or provided, and either (1) copies shall not be provided, or (2)
 24 if copies have already been provided, all copies in the receiving party's possession shall be promptly
 25 returned (and not relied upon) by the receiving party. Nothing in this paragraph shall prevent the
 26 receiving party from contending that the identified materials are not privileged, that the material was not
 27 inadvertently produced, or that privilege was waived for reasons other than mere inadvertent production
 28 of the material.

1
 2 22. Within ninety (90) days after conclusion of this litigation and any and all appeals thereof,
 3 any document and all reproductions of "Confidential" documents produced by a party that are in the
 4 possession of any Qualified Person shall be returned to the producing party or, with the consent of the
 producing party, destroyed. If destroyed, counsel for the receiving party shall certify to counsel for the
 producing party compliance with this paragraph within fourteen (14) calendar days of such destruction.
 Outside Counsel for each party may maintain in its files one copy of all material produced as well as all
 materials filed with or otherwise presented to the Court, deposition and trial transcripts, and work
 product¹⁰ (regardless of whether such materials contain or refer to "Confidential" materials). If counsel
 retains¹¹ such materials, the materials which contain Confidential Information shall be accessible only by
 Qualified Persons defined in paragraph 4(b) above. As far as the provisions of any protective orders
 entered¹³ in this action restrict the communication and use of the documents produced thereunder, such
 orders shall continue to be binding after the conclusion of this litigation including any subsequent
 appeals or later proceedings, except that (a) there shall be no restriction on documents that are used as
 exhibits¹⁴ in Court unless such exhibits were filed under seal, and (b) a party may seek the written
 permission¹⁶ of the producing party or order of the Court with respect to dissolution or modification of
 such protective orders. The Court shall retain jurisdiction to enforce the performance of said obligations.

20
 21 23. (a) At the election of the Producing Party, a Receiving Party's access to a Producing
 Party's²² discoverable source code may be limited to inspection of the code at a secured facility provided
 by the²³ Producing Party. Such inspection may be conducted only by persons identified in advance by the
 Receiving Party on a list of "Qualified Inspecting Personnel" which may include:

- 25 (1) the Receiving Party's Outside Counsel of record in this action; and
 26 (2) up to three Experts (as defined in this Order) of the Receiving Party to
 whom disclosure is reasonably necessary for this litigation and who have signed the
 27 "Agreement to Be Bound by Protective Order (Exhibit A) and who have been approved
 28 pursuant to the "Procedures for Approving Disclosure of 'CONFIDENTIAL' information

1
2 of Items to 'Experts'" as set forth in paragraph 5.
3

4 (b) The following provisions relate to Synopsys' provision of access to the Source
Code for versions of its commercial products:

5 (1) Synopsys will make available a closed room at its facility in Bethesda,
6 Maryland for use by Ricoh's Qualified Inspecting Personnel. The room will be set aside
7 for the exclusive use of Ricoh's Qualified Inspecting Personnel and will not be used by
8 Synopsys or any other party when Ricoh's Qualified Inspecting Personnel are not present.
9 The room will be available for a minimum of twelve weeks. After twelve weeks, and
10 after consultation with Ricoh, Synopsys may close the facility pursuant to the procedures
11 described in paragraph 4 below. If the facility is closed, Synopsys agrees to make the
12 source code available for inspection under similar procedures at another date prior to the
13 close of expert discovery.

14 (2) Synopsys will equip the closed room with a private phone if a phone jack
15 is already available in the room, a stand-alone, non-networked, computer and high-speed
16 printer. The computer will be loaded with copies of the source code to be produced and
17 utilities required to review the code. The source code shall include the code which is
18 used by Synopsys and no notes, comments, or any segments shall be removed before
19 being made available. The computer will be equipped with the text editors available in a
20 standard Unix distribution, suitable for use in editing the source code. Synopsys will
21 assist Ricoh in loading software that Ricoh may require for analysis of the source code.
22 The computer shall also be loaded with a complete distribution of the Synopsys software
23 that is fully operable and executable.

24 (3) Ricoh may print copies of a reasonable subset of the source code for the
25 Synopsys products at issue. Any printing done at the secure facility will be done
26 exclusively on paper supplied by Synopsys. Synopsys may elect to place preprinted
27 confidential designations on the margins of the paper. Ricoh's Qualified Inspecting
28 Personnel are not to bring blank paper into the closed room except for the purpose of
 making handwritten notes. Synopsys will initially supply Ricoh with 5,000 pages of
 paper for use with the printer. This figure is based on the estimate that 5,000 pages
 should be sufficient to print approximately 5% of the source code for Design Compiler. If
 at any time, Ricoh believes that additional printing and paper is required, Ricoh may
 submit additional requests for paper to Synopsys with a general statement of the basis of
 the request. Synopsys will respond within one week to any such request. If the parties

are unable to come to agreement after conferral, the matter may be presented to the Court. In evaluating requests for paper, the relevant standard to be applied is that Ricoh should be allowed to print hardcopies of a reasonable subset of the Synopsys source code and that what is reasonable shall be evaluated in light of relevance of the code to Ricoh's allegations and Synopsys' interest in preventing release in hardcopy of more than a fraction of its source code.

13 (c) The Receiving Party will provide the Producing Party with a copy of its list of
"Qualified Inspecting Personnel" no later than 5 business days before any person on the list attempts to
access the secured facility. The Receiving Party may revise the list to add or remove individuals,
provided that no more than a total of three Experts are ever provided with access to the source code
during the entire course of the litigation absent an agreement by the parties or a Court Order to expand
this number.

16 (d) Any notes taken or any other information created by Outside Counsel or the experts of the Receiving Party at or based on any inspection of the source code shall be treated as "CONFIDENTIAL" under this Protective Order.

18 24. This Order shall not bar any attorney herein in the course of rendering advice to his client
with respect to this litigation from conveying to any party client his evaluation in a general way of
19 "Confidential Information" produced or exchanged herein; provided, however, that in rendering such
20 advice and otherwise communicating with his client, the attorney shall not disclose the specific contents
21 of any 'Confidential Information" produced by another party herein, which disclosure would be contrary
22
23 to the terms of this Protective Order.

25. The Court shall retain jurisdiction to enforce the terms of this order for six (6) months
25 after the final termination of this action.
26

Dated: March 23, 2004

HOWREY SIMON ARNOLD & WHITE LLP

28

/s/ Christopher L. Kelley
Teresa M. Corbin, Esq.

1
2 Christopher Kelley, Esq.
3 Erik K. Moller, Esq.
4 Attorneys for Plaintiff SYNOPSYS, INC.
5 301 Ravenswood Avenue
6 Menlo Park, CA 94025
Telephone: (650) 463-8100
Facsimile: (650) 463-8400

7 Dated: March 23, 2004 DICKSTEIN SHAPIRO MORIN & OSHINSKY, LLP
8

9 /s/ Kenneth W. Brothers
10 Kenneth W. Brothers, Esq. (*pro hac vice*)
11 Attorneys for Defendant RICOH COMPANY, LTD.
12 2101 L Street NW
Washington, DC 20037
Telephone: (202) 785-9700
Facsimile: (202) 887-0689

13
14 ORDER

15 SO ORDERED this 24th day of March, 2004.
16

17 /s/
18 Magistrate Judge Edward M. Chen

19
20
21
22
23
24
25
26
27
28

1 Teresa M. Corbin (SBN 132360)
 2 Denise M. De Mory (SBN 168076)
 3 Ethan B. Andelman (SBN 209101)
 4 Jaclyn C. Fink (SBN 217913)
 5 HOWREY LLP
 6 525 Market Street, Suite 3600
 7 San Francisco, California 94105
 8 Telephone: (415) 848-4900
 9 Facsimile: (415) 848-4999

10 Attorneys for Plaintiff SYNOPSYS, INC.
 11 and for Defendants AEROFLEX INCORPORATED,
 12 AMI SEMICONDUCTOR, INC., MATROX
 13 ELECTRONIC SYSTEMS, LTD., MATROX
 14 GRAPHICS, INC., MATROX INTERNATIONAL
 15 CORP., MATROX TECH, INC., and
 16 AEROFLEX COLORADO SPRINGS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,

Case No. C03-04669 MJJ (EMC)

Plaintiff,

Case No. C03-02289 MJJ (EMC)

vs.

**NOTICE OF SUBPOENA TO YALE
UNIVERSITY**

AEROFLEX INCORPORATED, AMI
 SEMICONDUCTOR, INC., MATROX
 ELECTRONIC SYSTEMS LTD., MATROX
 GRAPHICS INC., MATROX INTERNATIONAL
 CORP., MATROX TECH, INC., AND
 AEROFLEX COLORADO SPRINGS, INC.

Defendants.

SYNOPSYS, INC.,

Plaintiff,

vs.

RICOH COMPANY, LTD.,

Defendant.

1 TO ALL PARTIES AND THEIR ATTORNEYS OF RECORD:

2 YOU ARE HEREBY NOTIFIED that pursuant to Federal Rule of Civil Procedure 45, Plaintiff
3 Synopsys, Inc. has served Yale University c/o Dorothy K. Robinson, Vice President and General Counsel,
4 the attached subpoena for production of documents.

5 The University is required to produce documents in its custody, possession or control specified in
6 Attachment A to the subpoena by 10:00 a.m. EST on Friday, August 18, 2006 at Legal Impressions, 205
7 Church Street, New Haven Connecticut, Telephone: (203) 907-4557.

8 Dated: August 14, 2006

HOWREY LLP

9
10 By: 

Matthew F. Greinert
Attorneys for Plaintiff SYNOPSYS
Defendants AEROFLEX
INCORPORATED, AMI
SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS, LTD.,
MATROX GRAPHICS, INC., MATROX
INTERNATIONAL CORP., MATROX
TECH, INC., and AEROFLEX
COLORADO SPRINGS, INC.

PROOF OF SERVICE

2 STATE OF CALIFORNIA)
3 COUNTY OF SAN FRANCISCO)
) ss.:

5 I am employed in the County of San Francisco, State of California. I am over the age of 18 and not
a party to the within action. My business address is 525 Market Street, Suite 3600, San Francisco,
California 94105.

On August 14, 2006 I served on the interested parties in said action the within:

NOTICE OF SUBPOENA TO YALE UNIVERSITY

8 by causing said document to be sent by Electronic Mail on August 14, 2006 to the email addresses
9 indicated for the parties listed below and by placing a true copy thereof in a sealed envelope(s) addressed as
stated below and causing such envelope(s) to be delivered as follows:

10 Gary M. Hoffman, Esq.
HoffmanG@dsmo.com
11 Dickstein Shapiro Morin & Oshinsky, LLP
2101 L Street, N.W.
12 Washington, DC 20037-1526

Jeffrey Demain, Esq.
jdemain@altshulerberzon.com
Altshuler, Berzon, Nussbaum, Rubin & Demain
177 Post Street, Suite 300
San Francisco, CA 94108

12 || Facsimile No.: (202) 887-0689

Facsimile No.: (415) 362-8064

14 Edward A. Meilman, Esq.
MeilmanE@dsmo.com
15 Dickstein Shapiro Morin & Oshinsky, LLP
1177 Avenue of the Americas
New York, NY 10036-2714

Facsimile No.: (212) 896-5471

- (OVERNIGHT DELIVERY) on August 14, 2006 by depositing in a box or other facility regularly maintained by Federal Express, an express service carrier, or delivering to a courier or driver authorized by said express service carrier to receive documents, a true copy of the foregoing document in sealed envelopes or packages designated by the express service carrier, addressed as stated above, with fees for overnight delivery paid or provided for and causing such envelope(s) to be delivered by said express service carrier on.

I declare under penalty of perjury that I am employed in the office of a member of the bar of this Court at whose direction the service was made and that the foregoing is true and correct.

Executed on August 14, 2006, at San Francisco, California.

James M. James
(Type or print name)

Jane M. Davis
(Signature)

Issued by the

UNITED STATES DISTRICT COURT
DISTRICT OF CONNECTICUT

RICOH COMPANY, LTD

V.

AEROFLEX, INCORPORATED, et al.

SUBPOENA IN A CIVIL CASE

Case Number: ¹ C03-04669 MJJ (EMC)
(Pending in the US District Court for
the Northern District of California)

TO: DOROTHY K. ROBINSON, Esq., Vice President & General Counsel,
on behalf of YALE UNIVERSITY
2 Whitney Avenue, 6th Floor
New Haven, Connecticut 06510

YOU ARE COMMANDED to appear in the United States District court at the place, date, and time specified below to testify in the above case.

PLACE OF TESTIMONY	COURTROOM
	DATE AND TIME

YOU ARE COMMANDED to appear at the place, date, and time specified below to testify at the taking of a deposition in the above case.

PLACE OF DEPOSITION	DATE AND TIME

YOU ARE COMMANDED to produce and permit inspection and copying of the following documents or objects at the place, date, and time specified below (list documents or objects):

SEE ATTACHMENT A

PLACE Legal Impressions, attention Michael Ferreira 205 Church St, New Haven, Connecticut, 203-907-4557	DATE AND TIME August 18, 2006, 10:00 a.m. EST
--	--

YOU ARE COMMANDED to permit inspection of the following premises at the date and time specified below.

PREMISES	DATE AND TIME

Any organization not a party to this suit that is subpoenaed for the taking of a deposition shall designate one or more officers, directors, or managing agents, or other persons who consent to testify on its behalf, and may set forth, for each person designated, the matters on which the person will testify. Federal Rules of Civil Procedure, 30(b)(6).

ISSUING OFFICER'S SIGNATURE AND TITLE (INDICATE IF ATTORNEY FOR PLAINTIFF OR DEFENDANT) Attorney for Defendant	DATE August 14, 2006
---	-------------------------

ISSUING OFFICER'S NAME, ADDRESS AND PHONE NUMBER

Matthew F. Greinert, HOWREY LLP, 525 Market Street, Suite 3600, San Francisco, CA 94105; Telephone: (415) 848-4900

(See Rule 45, Federal Rules of Civil Procedure, Parts C & D on next page)

¹ If action is pending in district other than district of issuance, state district under case number.

PROOF OF SERVICE

DATE _____ PLACE _____

SERVED:

SERVED ON (PRINT NAME) _____ MANNER OF SERVICE _____

SERVED BY (PRINT NAME) _____ TITLE _____

DECLARATION OF SERVER

I declare under penalty of perjury under the laws of the United States of America that the foregoing information contained in the Proof of Service is true and correct.

Executed on _____

DATE _____

SIGNATURE OF SERVER _____

ADDRESS OF SERVER _____

Rule 45, Federal Rules of Civil Procedure, Parts C & D:**(c) PROTECTION OF PERSONS SUBJECT TO SUBPOENAS.**

(1) A party or an attorney responsible for the issuance and service of a subpoena shall take reasonable steps to avoid imposing undue burden or expense on a person subject to that subpoena. The court on behalf of which the subpoena was issued shall enforce this duty and impose upon the party or attorney in breach of this duty an appropriate sanction which may include, but is not limited to, lost earnings and reasonable attorney's fee.

(2) (A) A person commanded to produce and permit inspection and copying of designated books, papers, documents or tangible things, or inspection of premises need not appear in person at the place of production or inspection unless commanded to appear for deposition, hearing or trial.

(B) Subject to paragraph (d) (2) of this rule, a person commanded to produce and permit inspection and copying may, within 14 days after service of subpoena or before the time specified for compliance if such time is less than 14 days after service, serve upon the party or attorney designated in the subpoena written objection to inspection or copying of any or all of the designated materials or of the premises. If objection is made, the party serving the subpoena shall not be entitled to inspect and copy materials or inspect the premises except pursuant to an order of the court by which the subpoena was issued. If objection has been made, the party serving the subpoena may, upon notice to the person commanded to produce, move at any time for an order to compel the production. Such an order to comply production shall protect any person who is not a party or an officer of a party from significant expense resulting from the inspection and copying commanded.

(3) (A) On timely motion, the court by which a subpoena was issued shall quash or modify the subpoena if it

(i) fails to allow reasonable time for compliance,

(ii) requires a person who is not a party or an officer of a party to travel to a place more than 100 miles from the place where that person resides, is employed or regularly transacts business in person, except that, subject to the provisions of clause (c) (3) (B) (iii) of this rule, such a person may in order to attend

trial be commanded to travel from any such place within the state in which the trial is held, or

(iii) requires disclosure of privileged or other protected matter and no exception or waiver applies, or

(iv) subjects a person to undue burden.

(B) If a subpoena

(i) requires disclosure of a trade secret or other confidential research, development, or commercial information, or

(ii) requires disclosure of an unretained expert's opinion or information not describing specific events or occurrences in dispute and resulting from the expert's study made not at the request of any party, or

(iii) requires a person who is not a party or an officer of a party to incur substantial expense to travel more than 100 miles to attend trial, the court may, to protect a person subject to or affected by the subpoena, quash or modify the subpoena, or, if the party in whose behalf the subpoena is issued shows a substantial need for the testimony or material that cannot be otherwise met without undue hardship and assures that the person to whom the subpoena is addressed will be reasonably compensated, the court may order appearance or production only upon specified conditions.

(d) DUTIES IN RESPONDING TO SUBPOENA.

(1) A person responding to a subpoena to produce documents shall produce them as they are kept in the usual course of business or shall organize and label them to correspond with the categories in the demand.

(2) When information subject to a subpoena is withheld on a claim that it is privileged or subject to protection as trial preparation materials, the claim shall be made expressly and shall be supported by a description of the nature of the documents, communications, or things not produced that is sufficient to enable the demanding party to contest the claim.

ATTACHMENT A

Pursuant to Federal Rule of Civil Procedure 45 and as directed in the subpoena attached hereto, you are to produce all documents and things within the scope of the following definitions and descriptions that are within your possession, custody, or control. A Protective Order has been entered in this case by the United States District Court for the Northern District of California and is attached as Attachment B. Included in the Protective Order are provisions for the protection of confidential information produced by a third party. With respect to documents and things withheld under a claim of privilege, you are required under Rule 45 to describe the nature of the documents and things withheld in a manner sufficient to enable the demanding party to contest the claims.

DEFINITIONS

1. The terms "you," and "your," mean, without limitation, Yale University, including without limitation all of its subsidiaries, parents, departments and affiliates, and all past or present directors, officers, agents, representatives, employees, students, consultants, attorneys, entities acting in joint-venture or partnership relationships with Yale University and others acting on behalf of Yale University.

2. As used herein, the word "document" means the original and each nonidentical copy of any written, printed, typed, recorded, computerized, electronic, taped, graphic, or other matter, in whatever form, whether in final or draft, including but not limited to all materials that constitute "writings," "recordings," "photographs," "source code" or "executable code" within the broadest meaning of Rule 1001 of the Federal Rules of Evidence and all materials that constitute "documents" within the broadest meaning of Rule 34 of the Federal Rules of Civil Procedure. The word "document" includes, without limitation, printed matter, electronic mail, materials stored on computer hard drives, diskettes, tapes, any other computer media, and any other information stored magnetically, optically or in any electronic medium and/or form.

3. As used herein, "person" means any individual, firm, partnership, corporation, proprietorship, association, governmental body, or any other organization or entity.

4. As used herein, "communication" includes, without limitation, communications by whatever means transmitted (i.e., whether oral, written, electronic or other methods used), as well as any note, memorandum or other record thereof.

5. The terms "regarding, referring or relating to" and "concerning" mean reflecting, concerning, containing, pertaining, referring, relating to, indicating, showing, describing, evidencing, discussing, mentioning, embodying or computing.

6. Whenever the singular is used, it shall also be taken to include the plural, and vice versa. Whenever the conjunctive is used, it shall also be taken to include the disjunctive, and vice versa.

INSTRUCTIONS

The following instructions apply to each of the requests for documents set forth herein:

1. Please produce entire documents, including, but not limited to, attachments, enclosures, cover letters, memoranda, and appendices.

2. Pursuant to Rule 26(e) of the Federal Rules of Civil Procedure, these requests for documents shall be deemed continuous up to and following the trial of this proceeding such that any documents or things requested herein which is either discovered by you or comes within your possession, custody or control subsequent to your initial responses hereto but prior to the final conclusion of this case should be produced in a supplemental response to these Document Requests immediately upon its discovery or receipt by you or your counsel.

3. If any document is withheld under a claim of privilege, in order that the Court and the parties may determine the validity of the claim of privilege, please provide a privilege log identifying each document withheld, including

- a. The type of document;
- b. The approximate date, and manner of recording, creating or otherwise preparing the document;
- c. The subject matter of the document;
- d. The name and organizational position of the person(s) who produced the document,

- e. The name and organizational position of the person(s) who received a copy of the document, or to whom the document was disclosed; and
- f. The claimed grounds on which the document is being withheld and facts sufficient to show the basis for each claim of privilege.

4. If you object to any part of a request for documents and refuse to produce documents responsive to that part, state your objection and respond to the remaining portion of that request. If you object to the scope or time period of a request for documents, state your objection and respond to the request for documents for the scope or time period you believe is appropriate.

5. Please produce all documents in the order in which they are kept in the ordinary course of business, and in their original file folders, binders, covers or containers, or facsimile thereof.

6. Any document bearing any changes, including, but not limited to, markings, handwritten notation, or other differences, that are not a part of the original text, or any reproduction thereof, is to be considered a separate document for purposes of responding to the following document requests. English translations of partial translations of foreign language documents should also be considered separate documents.

7. If a requested document is in a language other than English, please produce both the original and any existing English translation thereof.

8. If any of the following requests for documents cannot be responded to in full after exercising due diligence to secure the requested documents, please so state and respond to the extent possible, specifying your inability to respond to the remainder and stating whatever information you have regarding, referring or relating to the unanswered portions. If your response is qualified in any particular manner, set forth the details of such qualification.

9. Please produce hard copies of electronic records or produce computerized information in an intelligible format with a description of the system from which it was derived sufficient to permit rendering the materials intelligible.

DOCUMENT REQUESTS**Request No.1:**

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Computer Systems" in the Electrical Engineering and Computer Science Department at Yale University, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, an instructional RISC microprocessor implemented in Verilog, and any other demonstrative software and hardware created for the course.

Request No. 2:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Computer-Aided Design of Integrated Circuits" in the Electrical Engineering and Computer Science Department at Yale University, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 3:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of any course within the Electrical Engineering and Computer Science Department, or any other Department, at Yale University, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for any such course.

ATTACHMENT B

1

2

3

4

5

6

7

8

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA
SAN FRANCISCO DIVISION

10

11

12

SYNOPSYS, INC.,
Plaintiff
vs.
RICOH COMPANY, LTD.,
Defendant.

CASE NO. C03-02289-MJJ (EMC)
STIPULATED PROTECTIVE ORDER

18

19 1. All Confidential Information produced or exchanged in the course of this litigation shall
be used solely for the purpose of preparation and trial of this litigation and for no other purpose
whatsoever, and shall not be disclosed to any person except in accordance with the terms hereof.

22 2. "Confidential Information," as used herein, means any information of any type, kind or
character that is designated as "Confidential" by any of the supplying or receiving parties, whether it be
a document, information contained in a document, information revealed during a deposition,
information revealed in an interrogatory answer or otherwise. In designating information as
"Confidential," a party will make such designation only as to that information that it in good faith
believes contains "Confidential Information."

28 3. (a) "Confidential Information" includes, but is not limited to, (i) proprietary technical

1
 2 information and specifications, (ii) trade secrets (iii) confidential know-how, and (iv) proprietary
 3 business and financial information and any other non-public information, the disclosure of which is
 4 likely to have the effect of causing significant competitive harm to the disclosing party or party from
 5 which the information was obtained. Nothing in this paragraph shall be construed to limit the
 6 description of "Confidential Information" set forth in paragraph 2.

8 (b) Nothing shall be regarded as "Confidential Information" if it is information that:
 9 (i) is in the public domain at the time of disclosure, as evidenced by a written document;
 10 (ii) becomes part of the public domain through no fault of the other party, as evidenced
 11 by a written document;
 12 (iii) was in the receiving party's rightful and lawful possession at the time of disclosure,
 13 as evidenced by a written document; or
 14 (iv) is lawfully received by the receiving party from a third party at a later date without
 15 restriction as to disclosure, provided such third party has the right to make the disclosure to the
 16 receiving party.
 17

18 4. "Qualified Persons," as used herein means:
 19 (a) To the Court and its officers and staff, including court reporters;
 20 (b) Outside attorneys of record for the parties in this litigation and employees of such
 21 attorneys to whom it is necessary that the material be shown for purposes of this litigation;
 22 (c) Outside experts, consultants, advisors or investigators (collectively referred to
 23 hereafter as "experts") who have signed an undertaking pursuant to paragraph 5 but only after
 24 compliance with the provisions of paragraph 5 below;
 25 (d) To non-party support services including, but not limited to, court reporters, outside
 26 copy services, document imaging and database services, design services who have signed confidentiality
 27 agreements, jury consultants who have signed confidentiality agreements, mock jurors who have signed
 28 confidentiality agreements, and language translators who have signed confidentiality agreements

1
 2 (including support staff) as may be reasonably necessary in connection with the preparation or conduct
 3 of this action;
 4

5 (e) Anyone to whom the parties consent in writing; and
 6 (f) If this Court so elects, any other person may be designated as a Qualified Person by
 order of this Court, after notice and opportunity to be heard to all parties.

8 5. Prior to the disclosure of any "Confidential Information" to any expert under Paragraph
 4(c), counsel for the Party seeking to make the disclosure shall: (i) deliver a copy of this Protective
 Order as entered to such person, explain its terms to such person, and secure the signature of such
 10 person on a written undertaking in the form attached hereto as Exhibit A, and (ii) transmit by facsimile
 11 and mail to counsel for the other Parties a copy of the signed Exhibit A, accompanied by a curriculum
 12 vitae, at least ten (10) calendar days before any "Confidential Information" designated under this
 13 Protective Order is to be disclosed to the signator. The curriculum vitae should identify the general
 14 area(s) of expertise of the expert, provide a brief job history, specify all employment, expert or
 15 consulting engagements by the expert within the past five (5) years, and state all present or prior
 16 relationships between the expert and any entity directly or indirectly involved in this litigation or
 17 providing an indemnity to any such entity, its subsidiaries or its affiliates. Any Party may object to the
 18 proposed disclosure to an expert within the ten (10) calendar day period following the transmittal of
 19 Exhibit A and the curriculum vitae, by stating specifically in writing the reasons why the Party believes
 20 such expert should not receive designated "Confidential Information." If during that ten (10) calendar
 21 day period a Party makes such a written objection, there shall be no disclosure of "Confidential
 22 Information" to the expert absent mutual agreement of the Parties, waiver of the objection as stated
 23 below, or further order of the Court. After a Party objects to the proposed disclosure to an expert, the
 24 objecting Party shall move, by noticed motion or by ex parte application, for an order that disclosure not
 25 be made to such expert within five (5) business days following the date that the objection is made, or the
 26
 27
 28 Party's objection shall be deemed waived and disclosure may be made to the expert. The burden shall

1

2

be on the objecting Party to establish why the disclosure should not be made. Each Party shall maintain
3

a file of all such signed copies of Exhibit A. However, it shall not be necessary for administrative,
4

secretarial or clerical personnel working for such Qualified Person to sign a written undertaking.

6 6. (a) Documents produced in this action may be designated by any party or parties as
“Confidential” by marking each page of the document(s) with the designation “Confidential.”

8 (b) In lieu of marking the original of a document, if the original is not produced, the
designating party may mark the copies that are produced or exchanged. Originals shall be preserved for
inspection.
10

11 (c) If the document is not in paper form, the producing person or entity shall use other
12 such reasonable means as necessary to identify clearly the document or information as “Confidential.”

13 7. Discovery responses or other litigation materials may be designated by any party or
14 parties as “Confidential” by marking each page of the response with the designation “Confidential.”
15

16 8. The designation of information disclosed during a deposition as “Confidential” shall be
made either by a statement on the record at the deposition or within twenty (20) calendar days after
17 receipt by counsel of a copy of the deposition transcript. Such designation will be applied to only those
portions of the deposition transcript that include a specific question and response or series of questions
and responses containing “Confidential Information.” The deposition transcript shall be printed in
consecutive pages (whether or not some pages are designated as “Confidential”) with a marking on the
cover of the deposition transcript indicating the “Confidential” designation contained therein. Unless
previously designated otherwise, all deposition transcripts shall be treated as “Confidential” in their
entirety prior to the end of the twenty (20) calendar day period following receipt by counsel of a copy of
the deposition transcript.
25

26 9. “Confidential Information” shall not be disclosed or made available by the receiving
27 party to persons other than Qualified Persons except that nothing herein is intended to prevent
28 individuals who are in-house counsel or a member of the professional legal department of the Parties

1

2

from having access to pleadings, briefs and exhibits or declarations filed with the Court and expert
3
reports, including exhibits that are designated as "Confidential."
4

5 10. (a) Documents to be inspected shall be treated as "Confidential" although such documents need not be marked as "Confidential" prior to inspection. At the time of copying for the receiving parties, any documents containing "Confidential Information" shall be stamped prominently "Confidential" by the producing party.

9 (b) Nothing herein shall prevent disclosure beyond the terms of this Order if each party
10 designating the information as "Confidential" consents to such disclosure or if the Court, after notice to all effected parties, orders such disclosures. Nothing herein shall prevent any counsel of record from
11 utilizing "Confidential Information" in the examination or cross- examination of any person who is
12 indicated on the document as being an author, source or recipient of the "Confidential Information,"
13 irrespective of which party produced such information. Nothing herein shall prevent any counsel of
14 record from utilizing "Confidential Information" in the examination or cross-examination of any person
15 who is a current or former officer, director or employee of the party so designating the information as
16 "Confidential" or of the party that produced the information or of a related entity.
17
18

19 11. If a party inadvertently discloses any document or thing containing information that it deems confidential without designating it as "Confidential," the disclosing party shall promptly upon discovery of such inadvertent disclosure inform the receiving party in writing, and the receiving party and all
20 Qualified Persons possessing such information shall thereafter treat the information as
21 "Confidential" under this Order. To the extent such information may have been disclosed to persons
22 other than Qualified Persons described in this document, the receiving party shall make every
23 reasonable effort to retrieve the information promptly from such persons and to avoid any further
24 disclosure to and by such persons.
25
26

27 12. A party shall not be obligated to challenge the propriety of a designation as
28 "Confidential" at the time made, and a failure to do so shall not preclude a subsequent challenge thereto.

1

2

Nor will the failure to object be construed as an admission that any particular "Confidential
 3
 Information" contains or reflects currently valuable trade secrets or confidential commercial
 4
 information. In the event that any party to this litigation disagrees at any stage of these proceedings
 with the designation by the designating party of any information as "Confidential," or the designation of
 any person as a Qualified Person, the parties shall first try to resolve such dispute in good faith on an
 informa&Basis, such as production of redacted copies. If the parties are unsuccessful in informally
 resolving any disputes regarding the designation of any document or information as "Confidential," the
 Court shall resolve all such disputes. It shall be the burden of the party making any designation to
 establish that the information so designated is "Confidential" within the meaning of this Protective
 10 Order. The "Confidential Information" that is the subject of the dispute shall be treated as originally
 11 designated pending resolution of the dispute.

12

13. The parties may, by written stipulation filed and approved by the Court, amend this
 14 Order, and any party may seek an order of this Court modifying this Protective Order. The parties agree
 15 to meet and confer prior to seeking to modify this Protective Order. In addition, the Court may modify
 16 this Protective Order in the interest of justice or otherwise at the Court's discretion.
 17
 18

19 14. In the event a party wishes to use any "Confidential Information" in any affidavits,
 20 briefs, memoranda of law, or other papers filed with the Court in this litigation, such "Confidential
 21 Information" used therein shall be filed under seal with the Court. In addition to placing documents in a
 22 sealed envelope with instructions that the document is filed pursuant to the Stipulated Protective Order
 23 and that the envelope is not to be opened absent further order of the Court, the envelope should be
 24 labeled to identify the title of the case, the case number, and the title of the document.

25 15. The Clerk of this Court is directed to maintain under seal all documents and transcripts of
 26 deposition testimony and answers to interrogatories, admissions and other pleadings filed under seal
 27 with the Court in this litigation that have been designated, in whole or in part, as "Confidential" by a
 28 party to this action.

1
2 16. If a Party intends to offer into evidence or otherwise disclose in open court any
3 “Confidential Information” designated by another person or entity, counsel for such Party shall notify
4 the designating person or entity that the Party intends to disclose “Confidential Information” in open
court prior to the disclosure, so that the designating person or entity may confer with the Court
concerning appropriate procedures for protecting its “Confidential Information.”

8 17. In the event any person or party that has possession, custody, or control of any
information designated as “Confidential” pursuant to the terms of this Protective Order receives a
10 subpoena or other process or order to produce such information, such person or party shall notify by
mail within five (5) business days of the Party’s receipt of the request, the counsel for the party or
11 persons claiming confidential treatment of the documents sought by such subpoenas or other process or
12 order, shall furnish such counsel with a copy of said subpoena or other process or order, and shall
13 cooperate with respect to any procedure sought to be pursued by the party whose interests may be
14 affected. The party asserting the “Confidential” treatment shall have the burden of defending against
15 such subpoena, process or order. The person or party receiving the subpoena or process or order shall
16 be entitled to comply with it except: (a) to the extent the party asserting the “Confidential” treatment is
17 successful in obtaining an order modifying or quashing it; and (b) in complying with the process or
order shall, at a minimum, seek to obtain “Confidential” treatment of the “Confidential Information”
before producing it in the other proceeding or action.

22 18. If the discovery process calls for the production of information that a Party or Non-Party
does not wish to produce because the Party or Non-Party believes its disclosure would breach an
23 agreement with another person or entity to maintain such information in confidence, the disclosing Party
or Non-Party promptly shall give written notice to the other person or entity that its information is
24 subject to discovery in this litigation, and shall provide such person or entity with a copy of this
25 Protective Order. When such written notice is given to the person or entity, the disclosing Party or Non-
26 Party will advise the potential receiving Party that such notice has been given. The person or entity
27
28

1
 2 whose information may be subject to discovery shall have ten (10) business days from receipt of the
 3 written notice in which to seek relief from the Court, if the person or entity so desires. If the ten (10)
 4 business days elapse without the person or entity seeking relief from the Court, the requested
 5 information shall be produced in accordance with the terms of this Protective Order.

7 19. In the event that additional persons or entities become Parties, none of such Parties'
 8 counsel, experts or consultants retained to assist said counsel, shall have access to "Confidential
 9 Information" produced by or obtained from any other producing person or entity until said Party has
 10 executed and filed with the Court its agreement to be fully bound by this Protective Order.

11 20. This Protective Order shall apply to the parties and any non-party from whom discovery
 12 may be sought and who desires protection for the discovery sought. Thus, any non-party requested or
 13 required to produce or disclose information in this proceeding, through subpoena or otherwise, may
 14 designate such information pursuant to the terms of this Protective Order.

15 21. (a) Nothing herein requires disclosure of information, documents or things which the
 16 disclosing entity contends is protected from disclosure by the attorney-client privilege or the work-
 17 product exception. Nothing herein shall preclude any party from moving this Court for an order
 18 directing the disclosure of such information, documents or things.

20 (b) In the event that any privileged attorney-client or work product documents or things
 21 are inadvertently produced for inspection and/or provided, the disclosing party shall identify such
 22 documents or things within five (5) days of when it discovers that the privileged materials were
 23 inadvertently produced for inspection and/or provided, and either (1) copies shall not be provided, or (2)
 24 if copies have already been provided, all copies in the receiving party's possession shall be promptly
 25 returned (and not relied upon) by the receiving party. Nothing in this paragraph shall prevent the
 26 receiving party from contending that the identified materials are not privileged, that the material was not
 27 inadvertently produced, or that privilege was waived for reasons other than mere inadvertent production
 28 of the material.

1
 2 22. Within ninety (90) days after conclusion of this litigation and any and all appeals thereof,
 3 any document and all reproductions of "Confidential" documents produced by a party that are in the
 4 possession of any Qualified Person shall be returned to the producing party or, with the consent of the
 producing party, destroyed. If destroyed, counsel for the receiving party shall certify to counsel for the
 producing party compliance with this paragraph within fourteen (14) calendar days of such destruction.
 Outside & counsel for each party may maintain in its files one copy of all material produced as well as all
 material⁹ filed with or otherwise presented to the Court, deposition and trial transcripts, and work
 product¹⁰ (regardless of whether such materials contain or refer to "Confidential" materials). If counsel
 retains¹¹ such materials, the materials which contain Confidential Information shall be accessible only by
 Qualified Persons defined in paragraph 4(b) above. As far as the provisions of any protective orders
 entered¹² in this action restrict the communication and use of the documents produced thereunder, such
 orders shall continue to be binding after the conclusion of this litigation including any subsequent
 appeals or later proceedings, except that (a) there shall be no restriction on documents that are used as
 exhibits¹³ in Court unless such exhibits were filed under seal, and (b) a party may seek the written
 permission¹⁴ of the producing party or order of the Court with respect to dissolution or modification of
 such protective orders. The Court shall retain jurisdiction to enforce the performance of said obligations.
 15
 16
 17
 18

20

23. (a) At the election of the Producing Party, a Receiving Party's access to a Producing
 Party's²⁴ discoverable source code may be limited to inspection of the code at a secured facility provided
 by the Producing Party. Such inspection may be conducted only by persons identified in advance by the
 Receiving Party on a list of "Qualified Inspecting Personnel" which may include:
 25 (1) the Receiving Party's Outside Counsel of record in this action; and
 26 (2) up to three Experts (as defined in this Order) of the Receiving Party to
 27 whom disclosure is reasonably necessary for this litigation and who have signed the
 28 "Agreement to Be Bound by Protective Order (Exhibit A) and who have been approved
 pursuant to the "Procedures for Approving Disclosure of 'CONFIDENTIAL' information

1
2 of Items to 'Experts'" as set forth in paragraph 5.
3

4 (b) The following provisions relate to Synopsys' provision of access to the Source
Code for versions of its commercial products:

5 (1) Synopsys will make available a closed room at its facility in Bethesda,
6 Maryland for use by Ricoh's Qualified Inspecting Personnel. The room will be set aside
7 for the exclusive use of Ricoh's Qualified Inspecting Personnel and will not be used by
8 Synopsys or any other party when Ricoh's Qualified Inspecting Personnel are not present.
9 The room will be available for a minimum of twelve weeks. After twelve weeks, and
10 after consultation with Ricoh, Synopsys may close the facility pursuant to the procedures
11 described in paragraph 4 below. If the facility is closed, Synopsys agrees to make the
12 source code available for inspection under similar procedures at another date prior to the
13 close of expert discovery.

14 (2) Synopsys will equip the closed room with a private phone if a phone jack
15 is already available in the room, a stand-alone, non-networked, computer and high-speed
16 printer. The computer will be loaded with copies of the source code to be produced and
17 utilities required to review the code. The source code shall include the code which is
18 used by Synopsys and no notes, comments, or any segments shall be removed before
19 being made available. The computer will be equipped with the text editors available in a
20 standard Unix distribution, suitable for use in editing the source code. Synopsys will
21 assist Ricoh in loading software that Ricoh may require for analysis of the source code.
22 The computer shall also be loaded with a complete distribution of the Synopsys software
23 that is fully operable and executable.

24 (3) Ricoh may print copies of a reasonable subset of the source code for the
25 Synopsys products at issue. Any printing done at the secure facility will be done
26 exclusively on paper supplied by Synopsys. Synopsys may elect to place preprinted
27 confidential designations on the margins of the paper. Ricoh's Qualified Inspecting
28 Personnel are not to bring blank paper into the closed room except for the purpose of
29 making handwritten notes. Synopsys will initially supply Ricoh with 5,000 pages of
30 paper for use with the printer. This figure is based on the estimate that 5,000 pages
31 should be sufficient to print approximately 5% of the source code for Design Compiler. If
32 at any time, Ricoh believes that additional printing and paper is required, Ricoh may
33 submit additional requests for paper to Synopsys with a general statement of the basis of
34 the request. Synopsys will respond within one week to any such request. If the parties
35

are unable to come to agreement after conferral, the matter may be presented to the Court. In evaluating requests for paper, the relevant standard to be applied is that Ricoh should be allowed to print hardcopies of a reasonable subset of the Synopsys source code and that what is reasonable shall be evaluated in light of relevance of the code to Ricoh's allegations and Synopsys' interest in preventing release in hardcopy of more than a fraction of its source code.

13 (c) The Receiving Party will provide the Producing Party with a copy of its list of
"Qualified Inspecting Personnel" no later than 5 business days before any person on the list attempts to
access the secured facility. The Receiving Party may revise the list to add or remove individuals,
provided that no more than a total of three Experts are ever provided with access to the source code
during the entire course of the litigation absent an agreement by the parties or a Court Order to expand
this number.

16 (d) Any notes taken or any other information created by Outside Counsel or the experts of the Receiving Party at or based on any inspection of the source code shall be treated as "CONFIDENTIAL" under this Protective Order.

18 24. This Order shall not bar any attorney herein in the course of rendering advice to his client
with respect to this litigation from conveying to any party client his evaluation in a general way of
19 "Confidential Information" produced or exchanged herein; provided, however, that in rendering such
20 advice and otherwise communicating with his client, the attorney shall not disclose the specific contents
21 of any 'Confidential Information" produced by another party herein, which disclosure would be contrary
22
23 to the terms of this Protective Order.

24 25. The Court shall retain jurisdiction to enforce the terms of this order for six (6) months
25
after the final termination of this action.
26

Dated: March 23, 2004

HOWREY SIMON ARNOLD & WHITE LLP

28

/s/ Christopher L. Kelley
Teresa M. Corbin, Esq.

1 Teresa M. Corbin (SBN 132360)
2 Denise M. De Mory (SBN 168076)
3 Ethan B. Andelman (SBN 209101)
4 Jaclyn C. Fink (SBN 217913)
5 HOWREY LLP
 525 Market Street, Suite 3600
 San Francisco, California 94105
 Telephone: (415) 848-4900
 Facsimile: (415) 848-4999

6 Attorneys for Plaintiff SYNOPSYS, INC.
7 and for Defendants AEROFLEX INCORPORATED,
8 AMI SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS, LTD., MATROX
GRAPHICS, INC., MATROX INTERNATIONAL
CORP., MATROX TECH, INC., and
9 AEROFLEX COLORADO SPRINGS, INC.

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

SAN FRANCISCO DIVISION

RICOH COMPANY, LTD.,

Case No. C03-04669 MJJ (EMC)

Plaintiff.

Case No. C03-02289 MJJ (EMC)

VS.

AEROFLEX INCORPORATED, AMI
SEMICONDUCTOR, INC., MATROX
ELECTRONIC SYSTEMS LTD., MATROX
GRAPHICS INC., MATROX INTERNATIONAL
CORP., MATROX TECH, INC., AND
AEROFLEX COLORADO SPRINGS, INC.

**NOTICE OF SUBPOENA TO THE
MASSACHUSETTS INSTITUTE OF
TECHNOLOGY**

Defendants.

SYNOPSYS, INC.,

Plaintiff,

VS.

RICOH COMPANY, LTD.,

Defendant.

25

36

37

28

1 TO ALL PARTIES AND THEIR ATTORNEYS OF RECORD:

2 YOU ARE HEREBY NOTIFIED that pursuant to Federal Rule of Civil Procedure 45, Plaintiff

3 Synopsys, Inc. has served Massachusetts Institute of Technology c/o Jamie Lewis Keith, Senior Counsel,

4 the attached subpoena for production of documents.

5 The Massachusetts Institute of Technology is required to produce documents in its custody,

6 possession or control specified in Attachment A to the subpoena by 10:00 a.m. EST on Friday, August 18,

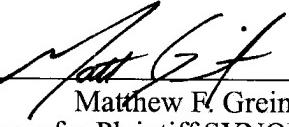
7 2006 at WarRoom Document Solutions, 274 Summer Street, 2nd Floor, Boston, Massachusetts 02210,

8 Telephone: (617) 426-6463.

9 Dated: August 14, 2006

HOWREY LLP

10 By: _____

11 
Matthew F. Greinert
12 Attorneys for Plaintiff SYNOPSYS
13 Defendants AEROFLEX
14 INCORPORATED, AMI
15 SEMICONDUCTOR, INC., MATROX
16 ELECTRONIC SYSTEMS, LTD.,
17 MATROX GRAPHICS, INC., MATROX
18 INTERNATIONAL CORP., MATROX
19 TECH, INC., and AEROFLEX
20 COLORADO SPRINGS, INC.

PROOF OF SERVICE

1 STATE OF CALIFORNIA)
 2) ss.:
 3 COUNTY OF SAN FRANCISCO)

4
 5 I am employed in the County of San Francisco, State of California. I am over the age of 18 and not
 6 a party to the within action. My business address is 525 Market Street, Suite 3600, San Francisco,
 7 California 94105.

8 On August 14, 2006 I served on the interested parties in said action the within:

**NOTICE OF SUBPOENA TO THE MASSACHUSETTS INSTITUTE OF
TECHNOLOGY**

9 by causing said document to be sent by Electronic Mail on August 14, 2006 to the email addresses
 10 indicated for the parties listed below and by placing a true copy thereof in a sealed envelope(s) addressed as
 11 stated below and causing such envelope(s) to be delivered as follows:

12 Gary M. Hoffman, Esq.
 13 HoffmanG@dsmo.com Jeffrey Demain, Esq.
 14 Dickstein Shapiro Morin & Oshinsky, LLP jdemain@altshulerberzon.com
 15 2101 L Street, N.W. Altshuler, Berzon, Nussbaum, Rubin & Demain
 16 Washington, DC 20037-1526 177 Post Street, Suite 300
 17 Facsimile No.: (202) 887-0689 San Francisco, CA 94108

Facsimile No.: (415) 362-8064

18 Edward A. Meilman, Esq.
 19 MeilmanE@dsmo.com
 20 Dickstein Shapiro Morin & Oshinsky, LLP
 21 1177 Avenue of the Americas
 22 New York, NY 10036-2714

23 Facsimile No.: (212) 896-5471

24 (OVERNIGHT DELIVERY) on August 14, 2006 by depositing in a box or other facility regularly
 25 maintained by Federal Express, an express service carrier, or delivering to a courier or driver
 26 authorized by said express service carrier to receive documents, a true copy of the foregoing
 27 document in sealed envelopes or packages designated by the express service carrier, addressed as
 28 stated above, with fees for overnight delivery paid or provided for and causing such envelope(s) to
 be delivered by said express service carrier on.

I declare under penalty of perjury that I am employed in the office of a member of the bar of this
 Court at whose direction the service was made and that the foregoing is true and correct.

Executed on August 14, 2006, at San Francisco, California.

James M. James
 (Type or print name)



(Signature)

Issued by the

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

RICOH COMPANY, LTD

V.
AEROFLEX, INCORPORATED, et al.

SUBPOENA IN A CIVIL CASE

Case Number: ¹ C03-04669 MJJ (EMC)
(Pending in the US District Court for
the Northern District of California)

TO: JAMIE LEWIS KEITH, Esq., Senior Counsel on behalf of
Massachusetts Institute of Technology
Office of the Senior Counsel, Room 7-206 & Room 12-090
77 Massachusetts Avenue, Cambridge, Massachusetts 02139

YOU ARE COMMANDED to appear in the United States District court at the place, date, and time specified below to testify in the above case.

PLACE OF TESTIMONY	COURTROOM
	DATE AND TIME

YOU ARE COMMANDED to appear at the place, date, and time specified below to testify at the taking of a deposition in the above case.

PLACE OF DEPOSITION	DATE AND TIME
---------------------	---------------

YOU ARE COMMANDED to produce and permit inspection and copying of the following documents or objects at the place, date, and time specified below (list documents or objects):

SEE ATTACHMENT A

PLACE WarRoom Document Solutions, attn: Errol Chin 274 Summer Street, 2 nd Floor, Boston, Massachusetts 02210, (617) 426-6463	DATE AND TIME August 18, 2006, 10:00 a.m. EST
---	--

YOU ARE COMMANDED to permit inspection of the following premises at the date and time specified below.

PREMISES	DATE AND TIME
----------	---------------

Any organization not a party to this suit that is subpoenaed for the taking of a deposition shall designate one or more officers, directors, or managing agents, or other persons who consent to testify on its behalf, and may set forth, for each person designated, the matters on which the person will testify. Federal Rules of Civil Procedure, 30(b)(6).

ISSUING OFFICER'S SIGNATURE AND TITLE (INDICATE IF ATTORNEY FOR PLAINTIFF OR DEFENDANT) Attorney for Defendant	DATE August 14, 2006
---	-------------------------

ISSUING OFFICER'S NAME, ADDRESS AND PHONE NUMBER

Matthew F. Greinert, HOWREY LLP, 525 Market Street, Suite 3600, San Francisco, CA 94105; Telephone: (415) 848-4900

(See Rule 45, Federal Rules of Civil Procedure, Parts C & D on next page)

¹ If action is pending in district other than district of issuance, state district under case number.

PROOF OF SERVICE

DATE _____ PLACE _____

SERVED:

SERVED ON (PRINT NAME) _____ MANNER OF SERVICE _____

SERVED BY (PRINT NAME) _____ TITLE _____

DECLARATION OF SERVER

I declare under penalty of perjury under the laws of the United States of America that the foregoing information contained in the Proof of Service is true and correct.

Executed on _____

DATE _____

SIGNATURE OF SERVER _____

ADDRESS OF SERVER _____

Rule 45, Federal Rules of Civil Procedure, Parts C & D:**(c) PROTECTION OF PERSONS SUBJECT TO SUBPOENAS.**

(1) A party or an attorney responsible for the issuance and service of a subpoena shall take reasonable steps to avoid imposing undue burden or expense on a person subject to that subpoena. The court on behalf of which the subpoena was issued shall enforce this duty and impose upon the party or attorney in breach of this duty an appropriate sanction which may include, but is not limited to, lost earnings and reasonable attorney's fee.

(2) (A) A person commanded to produce and permit inspection and copying of designated books, papers, documents or tangible things, or inspection of premises need not appear in person at the place of production or inspection unless commanded to appear for deposition, hearing or trial.

(B) Subject to paragraph (d) (2) of this rule, a person commanded to produce and permit inspection and copying may, within 14 days after service of subpoena or before the time specified for compliance if such time is less than 14 days after service, serve upon the party or attorney designated in the subpoena written objection to inspection or copying of any or all of the designated materials or of the premises. If objection is made, the party serving the subpoena shall not be entitled to inspect and copy materials or inspect the premises except pursuant to an order of the court by which the subpoena was issued. If objection has been made, the party serving the subpoena may, upon notice to the person commanded to produce, move at any time for an order to compel the production. Such an order to comply production shall protect any person who is not a party or an officer of a party from significant expense resulting from the inspection and copying commanded.

(3) (A) On timely motion, the court by which a subpoena was issued shall quash or modify the subpoena if it

- (i) fails to allow reasonable time for compliance,
- (ii) requires a person who is not a party or an officer of a party to travel to a place more than 100 miles from the place where that person resides, is employed or regularly transacts business in person, except that, subject to the provisions of clause (c) (3) (B) (iii) of this rule, such a person may in order to attend

trial be commanded to travel from any such place within the state in which the trial is held, or

(iii) requires disclosure of privileged or other protected matter and no exception or waiver applies, or

(iv) subjects a person to undue burden.

(B) If a subpoena

(i) requires disclosure of a trade secret or other confidential research, development, or commercial information, or

(ii) requires disclosure of an unretained expert's opinion or information not describing specific events or occurrences in dispute and resulting from the expert's study made not at the request of any party, or

(iii) requires a person who is not a party or an officer of a party to incur substantial expense to travel more than 100 miles to attend trial, the court may, to protect a person subject to or affected by the subpoena, quash or modify the subpoena, or, if the party in whose behalf the subpoena is issued shows a substantial need for the testimony or material that cannot be otherwise met without undue hardship and assures that the person to whom the subpoena is addressed will be reasonably compensated, the court may order appearance or production only upon specified conditions.

(d) DUTIES IN RESPONDING TO SUBPOENA.

(1) A person responding to a subpoena to produce documents shall produce them as they are kept in the usual course of business or shall organize and label them to correspond with the categories in the demand.

(2) When information subject to a subpoena is withheld on a claim that it is privileged or subject to protection as trial preparation materials, the claim shall be made expressly and shall be supported by a description of the nature of the documents, communications, or things not produced that is sufficient to enable the demanding party to contest the claim.

ATTACHMENT A

Pursuant to Federal Rule of Civil Procedure 45 and as directed in the subpoena attached hereto, you are to produce all documents and things within the scope of the following definitions and descriptions that are within your possession, custody, or control. A Protective Order has been entered in this case by the United States District Court for the Northern District of California and is attached as Attachment B. Included in the Protective Order are provisions for the protection of confidential information produced by a third party. With respect to documents and things withheld under a claim of privilege, you are required under Rule 45 to describe the nature of the documents and things withheld in a manner sufficient to enable the demanding party to contest the claims.

DEFINITIONS

1. The terms "you," and "your," mean, without limitation, Massachusetts Institute of Technology, including without limitation all of its subsidiaries, parents, departments and affiliates, and all past or present directors, officers, agents, representatives, employees, students, consultants, attorneys, entities acting in joint-venture or partnership relationships with Massachusetts Institute of Technology and others acting on behalf of Massachusetts Institute of Technology.

2. As used herein, the word "document" means the original and each nonidentical copy of any written, printed, typed, recorded, computerized, electronic, taped, graphic, or other matter, in whatever form, whether in final or draft, including but not limited to all materials that constitute "writings," "recordings," "photographs," "source code" or "executable code" within the broadest meaning of Rule 1001 of the Federal Rules of Evidence and all materials that constitute "documents" within the broadest meaning of Rule 34 of the Federal Rules of Civil Procedure. The word "document" includes, without limitation, printed matter, electronic mail, materials stored on computer hard drives, diskettes, tapes, any other computer media, and any other information stored magnetically, optically or in any electronic medium and/or form.

3. As used herein, "person" means any individual, firm, partnership, corporation, proprietorship, association, governmental body, or any other organization or entity.

4. As used herein, "communication" includes, without limitation, communications by whatever means transmitted (i.e., whether oral, written, electronic or other methods used), as well as any note, memorandum or other record thereof.

5. The terms "regarding, referring or relating to" and "concerning" mean reflecting, concerning, containing, pertaining, referring, relating to, indicating, showing, describing, evidencing, discussing, mentioning, embodying or computing.

6. Whenever the singular is used, it shall also be taken to include the plural, and vice versa. Whenever the conjunctive is used, it shall also be taken to include the disjunctive, and vice versa.

INSTRUCTIONS

The following instructions apply to each of the requests for documents set forth herein:

1. Please produce entire documents, including, but not limited to, attachments, enclosures, cover letters, memoranda, and appendices.

2. Pursuant to Rule 26(e) of the Federal Rules of Civil Procedure, these requests for documents shall be deemed continuous up to and following the trial of this proceeding such that any documents or things requested herein which is either discovered by you or comes within your possession, custody or control subsequent to your initial responses hereto but prior to the final conclusion of this case should be produced in a supplemental response to these Document Requests immediately upon its discovery or receipt by you or your counsel.

3. If any document is withheld under a claim of privilege, in order that the Court and the parties may determine the validity of the claim of privilege, please provide a privilege log identifying each document withheld, including

- a. The type of document;
- b. The approximate date, and manner of recording, creating or otherwise preparing the document;
- c. The subject matter of the document;

- d. The name and organizational position of the person(s) who produced the document,
- e. The name and organizational position of the person(s) who received a copy of the document, or to whom the document was disclosed; and
- f. The claimed grounds on which the document is being withheld and facts sufficient to show the basis for each claim of privilege.

4. If you object to any part of a request for documents and refuse to produce documents responsive to that part, state your objection and respond to the remaining portion of that request. If you object to the scope or time period of a request for documents, state your objection and respond to the request for documents for the scope or time period you believe is appropriate.

5. Please produce all documents in the order in which they are kept in the ordinary course of business, and in their original file folders, binders, covers or containers, or facsimile thereof.

6. Any document bearing any changes, including, but not limited to, markings, handwritten notation, or other differences, that are not a part of the original text, or any reproduction thereof, is to be considered a separate document for purposes of responding to the following document requests. English translations of partial translations of foreign language documents should also be considered separate documents.

7. If a requested document is in a language other than English, please produce both the original and any existing English translation thereof.

8. If any of the following requests for documents cannot be responded to in full after exercising due diligence to secure the requested documents, please so state and respond to the extent possible, specifying your inability to respond to the remainder and stating whatever information you have regarding, referring or relating to the unanswered portions. If your response is qualified in any particular manner, set forth the details of such qualification.

9. Please produce hard copies of electronic records or produce computerized information in an intelligible format with a description of the system from which it was derived sufficient to permit rendering the materials intelligible.

DOCUMENT REQUESTS

Request No.1:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Parallel Algorithms and Architectures" in the Electrical Engineering and Computer Science Department at Massachusetts Institute of Technology, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 2:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of a course titled "Advanced Algorithms" in the Electrical Engineering and Computer Science Department at Massachusetts Institute of Technology, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for the course.

Request No. 3:

All documents regarding, referring or relating to Dr. Marios Papaefthymiou's teaching of any course within the Electrical Engineering and Computer Science Department, or any other Department, at Massachusetts Institute of Technology, including but not limited to, course syllabi, handouts, outlines, digests, lecture notes, presentations, computer code, and demonstrative software and hardware created for any such course.

Request No. 4:

All documents regarding, referring or relating to theses authored by Dr. Marios Papaefthymiou, including but not limited to, MIT Laboratory for Computer Science Technical Reports TR-605 and TR-486, and any other technical bulletins or reports authored by Dr. Papaefthymious while he was a student at Massachusetts Institute of Technology.

ATTACHMENT B

1
2
3
4
5
6
7
8 UNITED STATES DISTRICT COURT
9 NORTHERN DISTRICT OF CALIFORNIA
10 SAN FRANCISCO DIVISION
11

12 SYNOPSYS, INC., } CASE NO. C03-02289-MJJ (EMC)
13 Plaintiff, }
14 vs. } STIPULATED PROTECTIVE ORDER
15 RICOH COMPANY, LTD., }
16 Defendant. }
17

18 1. All Confidential Information produced or exchanged in the course of this litigation shall
19 be used solely for the purpose of preparation and trial of this litigation and for no other purpose
20 whatsoever, and shall not be disclosed to any person except in accordance with the terms hereof.
21

22 2. "Confidential Information," as used herein, means any information of any type, kind or
23 character that is designated as "Confidential" by any of the supplying or receiving parties, whether it be
24 a document, information contained in a document, information revealed during a deposition,
25 information revealed in an interrogatory answer or otherwise. In designating information as
26 "Confidential," a party will make such designation only as to that information that it in good faith
27 believes contains "Confidential Information."

28 3. (a) "Confidential Information" includes, but is not limited to, (i) proprietary technical

1
 2 information and specifications, (ii) trade secrets (iii) confidential know-how, and (iv) proprietary
 3 business and financial information and any other non-public information, the disclosure of which is
 4 likely to have the effect of causing significant competitive harm to the disclosing party or party from
 5 which the information was obtained. Nothing in this paragraph shall be construed to limit the
 6 description of "Confidential Information" set forth in paragraph 2.

8 (b) Nothing shall be regarded as "Confidential Information" if it is information that:
 9 (i) is in the public domain at the time of disclosure, as evidenced by a written document;
 10 (ii) becomes part of the public domain through no fault of the other party, as evidenced
 11 by a written document;
 12 (iii) was in the receiving party's rightful and lawful possession at the time of disclosure,
 13 as evidenced by a written document; or
 14 (iv) is lawfully received by the receiving party from a third party at a later date without
 15 restriction as to disclosure, provided such third party has the right to make the disclosure to the
 16 receiving party.
 17

18 4. "Qualified Persons," as used herein means:
 19 (a) To the Court and its officers and staff, including court reporters;
 20 (b) Outside attorneys of record for the parties in this litigation and employees of such
 21 attorneys to whom it is necessary that the material be shown for purposes of this litigation;
 22 (c) Outside experts, consultants, advisors or investigators (collectively referred to
 23 hereafter as "experts") who have signed an undertaking pursuant to paragraph 5 but only after
 24 compliance with the provisions of paragraph 5 below;
 25 (d) To non-party support services including, but not limited to, court reporters, outside
 26 copy services, document imaging and database services, design services who have signed confidentiality
 27 agreements, jury consultants who have signed confidentiality agreements, mock jurors who have signed
 28 confidentiality agreements, and language translators who have signed confidentiality agreements

1
 2
 (including support staff) as may be reasonably necessary in connection with the preparation or conduct
 3
 of this action;
 4

5 (e) Anyone to whom the parties consent in writing; and
 6 (f) If this Court so elects, any other person may be designated as a Qualified Person by
 order of this Court, after notice and opportunity to be heard to all parties.

8 5. Prior to the disclosure of any "Confidential Information" to any expert under Paragraph
 4(c), counsel for the Party seeking to make the disclosure shall: (i) deliver a copy of this Protective
 Order as entered to such person, explain its terms to such person, and secure the signature of such
 11 person on a written undertaking in the form attached hereto as Exhibit A, and (ii) transmit by facsimile
 and mail to counsel for the other Parties a copy of the signed Exhibit A, accompanied by a curriculum
 12 vitae, at least ten (10) calendar days before any "Confidential Information" designated under this
 13 Protective Order is to be disclosed to the signator. The curriculum vitae should identify the general
 14 area(s) of expertise of the expert, provide a brief job history, specify all employment, expert or
 15 consulting engagements by the expert within the past five (5) years, and state all present or prior
 16 relationships between the expert and any entity directly or indirectly involved in this litigation or
 17 providing an indemnity to any such entity, its subsidiaries or its affiliates. Any Party may object to the
 18 proposed disclosure to an expert within the ten (10) calendar day period following the transmittal of
 19 Exhibit A and the curriculum vitae, by stating specifically in writing the reasons why the Party believes
 20 such expert should not receive designated "Confidential Information." If during that ten (10) calendar
 21 day period a Party makes such a written objection, there shall be no disclosure of "Confidential
 22 Information" to the expert absent mutual agreement of the Parties, waiver of the objection as stated
 23 below, or further order of the Court. After a Party objects to the proposed disclosure to an expert, the
 24 objecting Party shall move, by noticed motion or by ex parte application, for an order that disclosure not
 25 be made to such expert within five (5) business days following the date that the objection is made, or the
 26
 27 Party's objection shall be deemed waived and disclosure may be made to the expert. The burden shall
 28

1
 2
 be on the objecting Party to establish why the disclosure should not be made. Each Party shall maintain
 3
 a file of all such signed copies of Exhibit A. However, it shall not be necessary for administrative,
 4
 secretarial or clerical personnel working for such Qualified Person to sign a written undertaking.

6 6. (a) Documents produced in this action may be designated by any party or parties as
 "Confidential" by marking each page of the document(s) with the designation "Confidential."

8 (b) In lieu of marking the original of a document, if the original is not produced, the
 designating party may mark the copies that are produced or exchanged. Originals shall be preserved for
 inspection.

11 (c) If the document is not in paper form, the producing person or entity shall use other
 12 such reasonable means as necessary to identify clearly the document or information as "Confidential."

13 7. Discovery responses or other litigation materials may be designated by any party or
 14 parties as "Confidential" by marking each page of the response with the designation "Confidential."

15 8. The designation of information disclosed during a deposition as "Confidential" shall be
 16 made either by a statement on the record at the deposition or within twenty (20) calendar days after
 17 receipt by counsel of a copy of the deposition transcript. Such designation will be applied to only those
 18 portions of the deposition transcript that include a specific question and response or series of questions
 and responses containing "Confidential Information." The deposition transcript shall be printed in
 consecutive pages (whether or not some pages are designated as "Confidential") with a marking on the
 cover of the deposition transcript indicating the "Confidential" designation contained therein. Unless
 previously designated otherwise, all deposition transcripts shall be treated as "Confidential" in their
 entirety prior to the end of the twenty (20) calendar day period following receipt by counsel of a copy of
 24 the deposition transcript.

26 9. "Confidential Information" shall not be disclosed or made available by the receiving
 27 party to persons other than Qualified Persons except that nothing herein is intended to prevent
 28 individuals who are in-house counsel or a member of the professional legal department of the Parties

1
2 from having access to pleadings, briefs and exhibits or declarations filed with the Court and expert
3 reports, including exhibits that are designated as "Confidential."
4

5 10. (a) Documents to be inspected shall be treated as "Confidential" although such documents need not be marked as "Confidential" prior to inspection. At the time of copying for the receiving parties, any documents containing "Confidential Information" shall be stamped prominently "Confidential" by the producing party.

9 (b) Nothing herein shall prevent disclosure beyond the terms of this Order if each party designating the information as "Confidential" consents to such disclosure or if the Court, after notice to all effected parties, orders such disclosures. Nothing herein shall prevent any counsel of record from utilizing "Confidential Information" in the examination or cross-examination of any person who is indicated on the document as being an author, source or recipient of the "Confidential Information," irrespective of which party produced such information. Nothing herein shall prevent any counsel of record from utilizing "Confidential Information" in the examination or cross-examination of any person who is a current or former officer, director or employee of the party so designating the information as "Confidential" or of the party that produced the information or of a related entity.
10
11
12
13
14
15
16
17
18

19 11. If a party inadvertently discloses any document or thing containing information that it deems confidential without designating it as "Confidential," the disclosing party shall promptly upon discovery of such inadvertent disclosure inform the receiving party in writing, and the receiving party and all Qualified Persons possessing such information shall thereafter treat the information as "Confidential" under this Order. To the extent such information may have been disclosed to persons other than Qualified Persons described in this document, the receiving party shall make every reasonable effort to retrieve the information promptly from such persons and to avoid any further disclosure to and by such persons.
20
21
22
23
24
25
26
27

12. A party shall not be obligated to challenge the propriety of a designation as "Confidential" at the time made, and a failure to do so shall not preclude a subsequent challenge thereto.
28

1
 2
 3 Nor will the failure to object be construed as an admission that any particular "Confidential
 4 Information" contains or reflects currently valuable trade secrets or confidential commercial
 information. In the event that any party to this litigation disagrees at any stage of these proceedings
 with the designation by the designating party of any information as "Confidential," or the designation of
 any person as a Qualified Person, the parties shall first try to resolve such dispute in good faith on an
 informa&Basis, such as production of redacted copies. If the parties are unsuccessful in informally
 resolving any disputes regarding the designation of any document or information as "Confidential," the
 Court shall resolve all such disputes. It shall be the burden of the party making any designation to
 establish that the information so designated is "Confidential" within the meaning of this Protective
 Order. The "Confidential Information" that is the subject of the dispute shall be treated as originally
 13 designated pending resolution of the dispute.

14 13. The parties may, by written stipulation filed and approved by the Court, amend this
 15 Order, and any party may seek an order of this Court modifying this Protective Order. The parties agree
 16 to meet and confer prior to seeking to modify this Protective Order. In addition, the Court may modify
 17 this Protective Order in the interest of justice or otherwise at the Court's discretion.
 18

19 14. In the event a party wishes to use any "Confidential Information" in any affidavits,
 20 briefs, memoranda of law, or other papers filed with the Court in this litigation, such "Confidential
 21 Information" used therein shall be filed under seal with the Court. In addition to placing documents in a
 22 sealed envelope with instructions that the document is filed pursuant to the Stipulated Protective Order
 23 and that the envelope is not to be opened absent further order of the Court, the envelope should be
 24 labeled to identify the title of the case, the case number, and the title of the document.

25 15. The Clerk of this Court is directed to maintain under seal all documents and transcripts of
 26 deposition testimony and answers to interrogatories, admissions and other pleadings filed under seal
 27 with the Court in this litigation that have been designated, in whole or in part, as "Confidential" by a
 28 party to this action.

1
 2 16. If a Party intends to offer into evidence or otherwise disclose in open court any
 3 “Confidential Information” designated by another person or entity, counsel for such Party shall notify
 4 the designating person or entity that the Party intends to disclose “Confidential Information” in open
 5 court prior to the disclosure, so that the designating person or entity may confer with the Court
 6 concerning appropriate procedures for protecting its “Confidential Information.”

8 17. In the event any person or party that has possession, custody, or control of any
 9 information designated as “Confidential” pursuant to the terms of this Protective Order receives a
 10 subpoena or other process or order to produce such information, such person or party shall notify by
 11 mail within five (5) business days of the Party’s receipt of the request, the counsel for the party or
 12 persons claiming confidential treatment of the documents sought by such subpoenas or other process or
 13 order, shall furnish such counsel with a copy of said subpoena or other process or order, and shall
 14 cooperate with respect to any procedure sought to be pursued by the party whose interests may be
 15 affected. The party asserting the “Confidential” treatment shall have the burden of defending against
 16 such subpoena, process or order. The person or party receiving the subpoena or process or order shall
 17 be entitled to comply with it except: (a) to the extent the party asserting the “Confidential” treatment is
 18 successful in obtaining an order modifying or quashing it; and (b) in complying with the process or
 19 order shall, at a minimum, seek to obtain “Confidential” treatment of the “Confidential Information”
 20 before producing it in the other proceeding or action.

22 18. If the discovery process calls for the production of information that a Party or Non-Party
 23 does not wish to produce because the Party or Non-Party believes its disclosure would breach an
 24 agreement with another person or entity to maintain such information in confidence, the disclosing Party
 25 or Non-Party promptly shall give written notice to the other person or entity that its information is
 26 subject to discovery in this litigation, and shall provide such person or entity with a copy of this
 27 Protective Order. When such written notice is given to the person or entity, the disclosing Party or Non-
 28 Party will advise the potential receiving Party that such notice has been given. The person or entity

1

2

whose information may be subject to discovery shall have ten (10) business days from receipt of the
 written notice in which to seek relief from the Court, if the person or entity so desires. If the ten (10)
 business days elapse without the person or entity seeking relief from the Court, the requested
 information shall be produced in accordance with the terms of this Protective Order.

7 19. In the event that additional persons or entities become Parties, none of such Parties'
 counsel, experts or consultants retained to assist said counsel, shall have access to "Confidential
 Information" produced by or obtained from any other producing person or entity until said Party has
 executed¹⁰ and filed with the Court its agreement to be fully bound by this Protective Order.

11 20. This Protective Order shall apply to the parties and any non-party from whom discovery
 may be sought and who desires protection for the discovery sought. Thus, any non-party requested or
 required to produce or disclose information in this proceeding, through subpoena or otherwise, may
 designate such information pursuant to the terms of this Protective Order.
 12
 13
 14
 15

21. (a) Nothing herein requires disclosure of information, documents or things which the
 disclosing entity contends is protected from disclosure by the attorney-client privilege or the work-
 product exception. Nothing herein shall preclude any party from moving this Court for an order
 directing¹⁷ the disclosure of such information, documents or things.
 16
 17
 18
 19

20 (b) In the event that any privileged attorney-client or work product documents or things
 are inadvertently produced for inspection and/or provided, the disclosing party shall identify such
 documents or things within five (5) days of when it discovers that the privileged materials were
 inadvertently produced for inspection and/or provided, and either (1) copies shall not be provided, or (2)
 if copies have already been provided, all copies in the receiving party's possession shall be promptly
 returned²³ (and not relied upon) by the receiving party. Nothing in this paragraph shall prevent the
 receiving party from contending that the identified materials are not privileged, that the material was not
 inadvertently produced, or that privilege was waived for reasons other than mere inadvertent production
 of the material.
 21
 22
 23
 24
 25
 26
 27
 28

1
 2 22. Within ninety (90) days after conclusion of this litigation and any and all appeals thereof,
 3 any document and all reproductions of "Confidential" documents produced by a party that are in the
 4 possession of any Qualified Person shall be returned to the producing party or, with the consent of the
 producing party, destroyed. If destroyed, counsel for the receiving party shall certify to counsel for the
 producing party compliance with this paragraph within fourteen (14) calendar days of such destruction.
 Outside Counsel for each party may maintain in its files one copy of all material produced as well as all
 material filed with or otherwise presented to the Court, deposition and trial transcripts, and work
 product¹⁰ (regardless of whether such materials contain or refer to "Confidential" materials). If counsel
 retains¹¹ such materials, the materials which contain Confidential Information shall be accessible only by
 Qualified Persons defined in paragraph 4(b) above. As far as the provisions of any protective orders
 entered in this action restrict the communication and use of the documents produced thereunder, such
 orders shall continue to be binding after the conclusion of this litigation including any subsequent
 appeals or later proceedings, except that (a) there shall be no restriction on documents that are used as
 exhibits in Court unless such exhibits were filed under seal, and (b) a party may seek the written
 permission of the producing party or order of the Court with respect to dissolution or modification of
 such protective orders. The Court shall retain jurisdiction to enforce the performance of said obligations.

20
 21 23. (a) At the election of the Producing Party, a Receiving Party's access to a Producing
 Party's²² discoverable source code may be limited to inspection of the code at a secured facility provided
 by the Producing Party. Such inspection may be conducted only by persons identified in advance by the
 Receiving Party on a list of "Qualified Inspecting Personnel" which may include:

- 25 (1) the Receiving Party's Outside Counsel of record in this action; and
- 26 (2) up to three Experts (as defined in this Order) of the Receiving Party to
 whom disclosure is reasonably necessary for this litigation and who have signed the
 27 "Agreement to Be Bound by Protective Order (Exhibit A) and who have been approved
 28 pursuant to the "Procedures for Approving Disclosure of 'CONFIDENTIAL' information

1
2 of Items to 'Experts'" as set forth in paragraph 5.
3 (b) The following provisions relate to Synopsys' provision of access to the Source
4 Code for versions of its commercial products:

5 (1) Synopsys will make available a closed room at its facility in Bethesda,
6 Maryland for use by Ricoh's Qualified Inspecting Personnel. The room will be set aside
7 for the exclusive use of Ricoh's Qualified Inspecting Personnel and will not be used by
8 Synopsys or any other party when Ricoh's Qualified Inspecting Personnel are not present.
9 The room will be available for a minimum of twelve weeks. After twelve weeks, and
10 after consultation with Ricoh, Synopsys may close the facility pursuant to the procedures
11 described in paragraph 4 below. If the facility is closed, Synopsys agrees to make the
12 source code available for inspection under similar procedures at another date prior to the
13 close of expert discovery.

14 (2) Synopsys will equip the closed room with a private phone if a phone jack
15 is already available in the room, a stand-alone, non-networked, computer and high-speed
16 printer. The computer will be loaded with copies of the source code to be produced and
17 utilities required to review the code. The source code shall include the code which is
18 used by Synopsys and no notes, comments, or any segments shall be removed before
19 being made available. The computer will be equipped with the text editors available in a
20 standard Unix distribution, suitable for use in editing the source code. Synopsys will
21 assist Ricoh in loading software that Ricoh may require for analysis of the source code.
22 The computer shall also be loaded with a complete distribution of the Synopsys software
23 that is fully operable and executable.

24 (3) Ricoh may print copies of a reasonable subset of the source code for the
25 Synopsys products at issue. Any printing done at the secure facility will be done
26 exclusively on paper supplied by Synopsys. Synopsys may elect to place preprinted
27 confidential designations on the margins of the paper. Ricoh's Qualified Inspecting
28 Personnel are not to bring blank paper into the closed room except for the purpose of
 making handwritten notes. Synopsys will initially supply Ricoh with 5,000 pages of
 paper for use with the printer. This figure is based on the estimate that 5,000 pages
 should be sufficient to print approximately 5% of the source code for Design Compiler. If
 at any time, Ricoh believes that additional printing and paper is required, Ricoh may
 submit additional requests for paper to Synopsys with a general statement of the basis of
 the request. Synopsys will respond within one week to any such request. If the parties

1
2 are unable to come to agreement after conferral, the matter may be presented to the
3 Court. In evaluating requests for paper, the relevant standard to be applied is that Ricoh
4 should be allowed to print hardcopies of a reasonable subset of the Synopsys source code
5 and that what is reasonable shall be evaluated in light of relevance of the code to Ricoh's
6 allegations and Synopsys' interest in preventing release in hardcopy of more than a
7 fraction of its source code.

13 (c) The Receiving Party will provide the Producing Party with a copy of its list of
"Qualified Inspecting Personnel" no later than 5 business days before any person on the list attempts to
access the secured facility. The Receiving Party may revise the list to add or remove individuals,
provided that no more than a total of three Experts are ever provided with access to the source code
during the entire course of the litigation absent an agreement by the parties or a Court Order to expand
this number.

16 (d) Any notes taken or any other information created by Outside Counsel or the experts of the Receiving Party at or based on any inspection of the source code shall be treated as "CONFIDENTIAL" under this Protective Order.

18 24. This Order shall not bar any attorney herein in the course of rendering advice to his client
with respect to this litigation from conveying to any party client his evaluation in a general way of
19 "Confidential Information" produced or exchanged herein; provided, however, that in rendering such
20 advice and otherwise communicating with his client, the attorney shall not disclose the specific contents
21 of any "Confidential Information" produced by another party herein, which disclosure would be contrary
22
23 to the terms of this Protective Order.

25. The Court shall retain jurisdiction to enforce the terms of this order for six (6) months
25 after the final termination of this action.
26

Dated: March 23, 2004

HOWREY SIMON ARNOLD & WHITE, L.L.P.

28

/s/ Christopher L. Kelley
Teresa M. Corbin, Esq.

1
2 Christopher Kelley, Esq.
3 Erik K. Moller, Esq.
4 Attorneys for Plaintiff SYNOPSYS, INC.
5 301 Ravenswood Avenue
6 Menlo Park, CA 94025
Telephone: (650) 463-8100
Facsimile: (650) 463-8400

7 Dated: March 23, 2004 DICKSTEIN SHAPIRO MORIN & OSHINSKY, LLP
8

9 /s/ Kenneth W. Brothers
10 Kenneth W. Brothers, Esq. (*pro hac vice*)
11 Attorneys for Defendant RICOH COMPANY, LTD.
12 2101 L Street NW
Washington, DC 20037
Telephone: (202) 785-9700
Facsimile: (202) 887-0689

13
14 ORDER

15 SO ORDERED this 24th day of March, 2004.
16

17 /s/
18 Magistrate Judge Edward M. Chen

19
20
21
22
23
24
25
26
27
28



US006226776B1

(12) **United States Patent**
Panchul et al.

(10) Patent No.: US 6,226,776 B1
(45) Date of Patent: May 1, 2001

(54) **SYSTEM FOR CONVERTING HARDWARE DESIGNS IN HIGH-LEVEL PROGRAMMING LANGUAGE TO HARDWARE IMPLEMENTATIONS**

(75) Inventors: Yuri V. Panchul, Milpitas; Donald A. Soderman, Saratoga; Denis R. Coleman, Atherton, all of CA (US)

(73) Assignee: Syntropy Corporation, Milpitas, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 08/931,148

(22) Filed: Sep. 16, 1997

(51) Int. Cl.⁷ G06F 17/50

(52) U.S. Cl. 716/3; 716/18

(58) Field of Search 395/500.04, 500.19; 716/3, 18

(56) **References Cited**

U.S. PATENT DOCUMENTS

- 5,197,016 * 3/1993 Sugimoto et al. 395/500.19
- 5,437,037 * 7/1995 Furuichi 395/700
- 5,493,507 * 2/1996 Shinde et al. 395/489
- 5,530,843 * 6/1996 Koyama 395/500.04
- 5,603,043 * 2/1997 Taylor et al. 395/800
- 5,748,488 * 5/1998 Gregory et al. 395/500.19

5,953,519 * 9/1999 Fura 395/500.19

* cited by examiner

Primary Examiner—Matthew Smith

Assistant Examiner—Leigh Marie Garbowsky

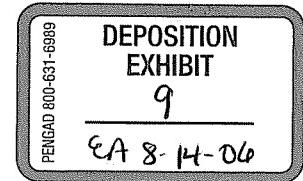
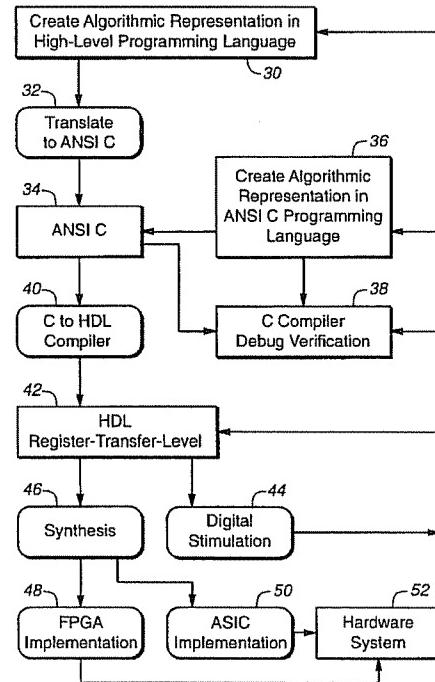
(74) Attorney, Agent, or Firm—Russo & Hale LLP; William C. Milks, III

(57) **ABSTRACT**

A computer aided hardware design system for enabling design of an actual hardware implementation for a digital circuit using a high-level algorithmic programming language. The system converts an algorithmic representation for a hardware design initially created in the high-level programming language, such as ANSI C, to a hardware design implementation, such as an FPGA or other programmable logic or an ASIC. The C-type program representative of the hardware design is compiled into a register transfer level (RTL) hardware description language (HDL) that can be synthesized into a gate-level hardware representation. The system additionally enables simulation of the HDL design to verify design functionality. Finally, various physical design tools can be utilized to produce an actual hardware implementation. The system also permits the use of other non-C-type high-level programming languages by first translating to a C-type program. In contrast to previous hardware design tools, the system compiles all C-type programming language features, including pointers and structures, into synthesizable HDL.

54 Claims, 90 Drawing Sheets

Microfiche Appendix Included
(2 Microfiche, 176 Pages)



U.S. Patent

May 1, 2001

Sheet 1 of 90

US 6,226,776 B1

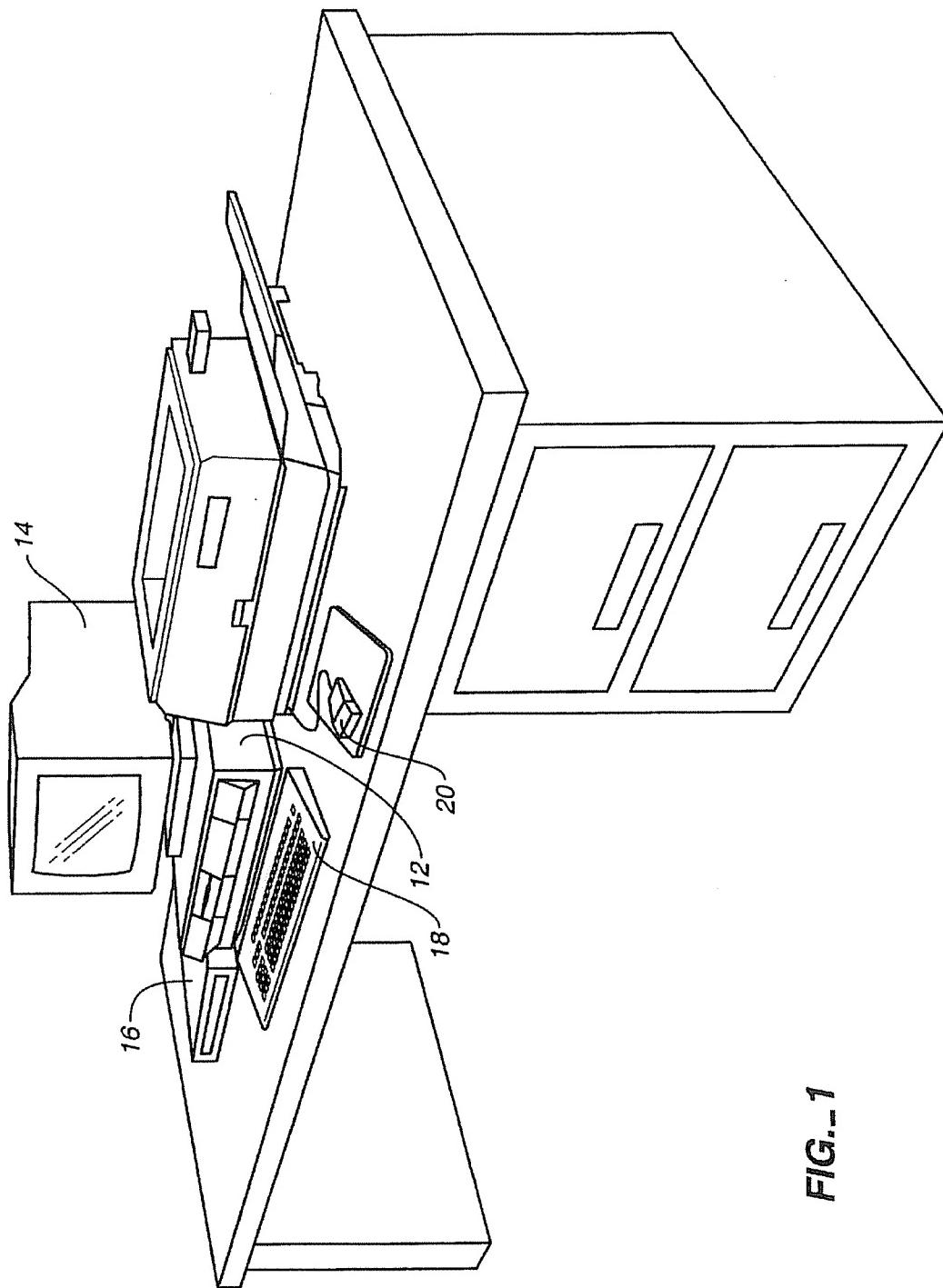


FIG.-1

U.S. Patent

May 1, 2001

Sheet 2 of 90

US 6,226,776 B1

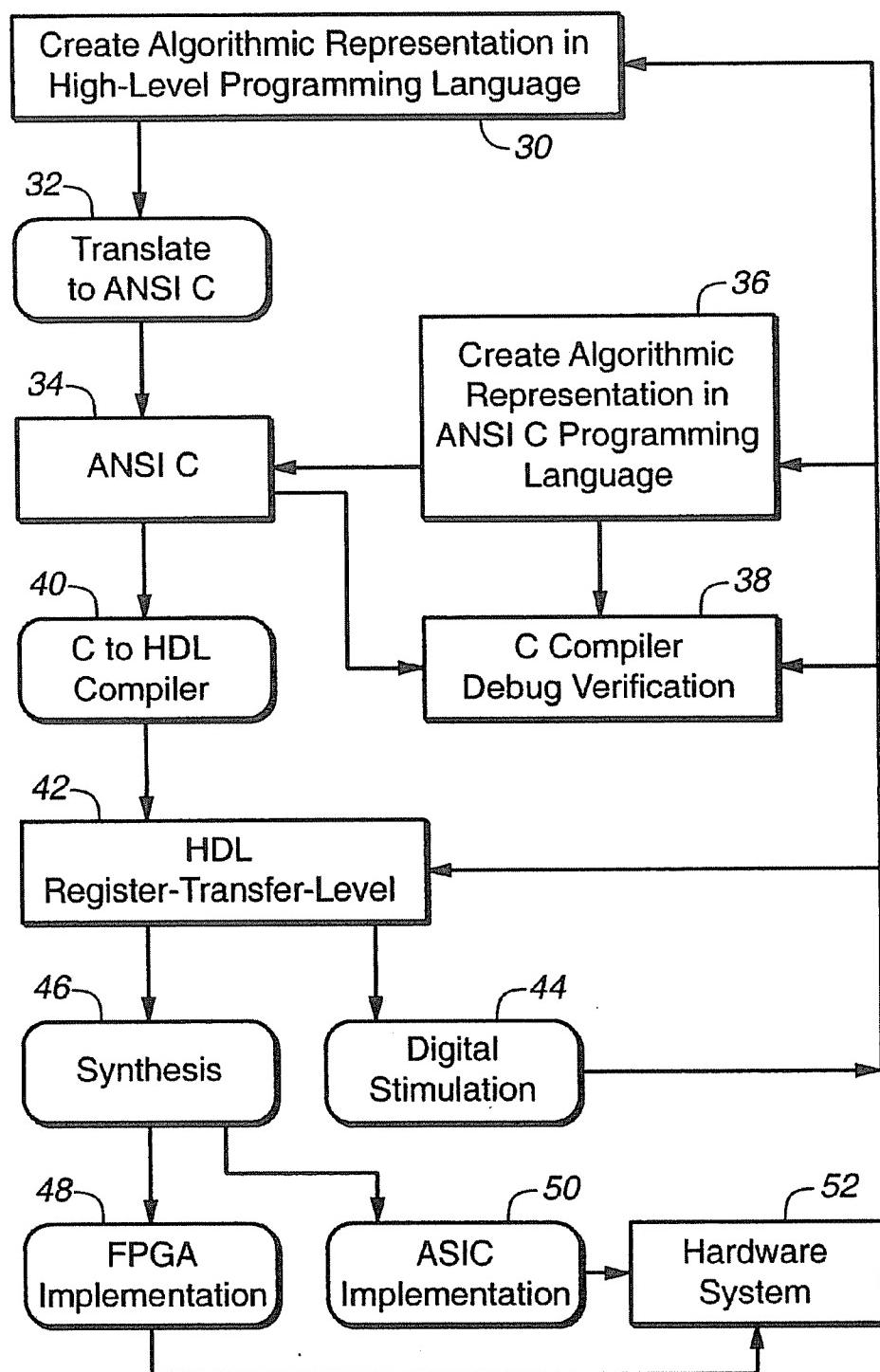


FIG._2

U.S. Patent May 1, 2001 Sheet 3 of 90 US 6,226,776 B1

```
int func1 (unsigned a, unsigned b, unsigned c)
{
    return a + b ^ c << 3;
}

int func2 (int a, int b, int c)
{
    return (a ^ b) & c;
}
```

FIG._3A

```
module func1(result_func1, result_func2, param_func1_a, param_func1_b,
            param_func1_c, param_func2_a, param_func2_b, param_func2_c);

    output [15:0]    result_func1, result_func2;
    input  [15:0]   param_func1_a, param_func1_b, param_func1_c;
    input  [15:0]   param_func2_a, param_func2_b, param_func2_c;

    wire  [15:0]    result_func1 =
                    ((param_func1_a + param_func1_b) ^ (param_func1_c << 16'd3));
    wire  [15:0]    result_func2 =
                    ((param_func2_a ^ param_func2_b) & param_func2_c);

endmodule
```

FIG._3B

U.S. Patent May 1, 2001 Sheet 4 of 90 US 6,226,776 B1

```
int func1 (unsigned a, unsigned b, unsigned c)
{
    return a + (b == 1) ^ c << 3;
}

int func2 (int a, int b, int c)
{
    if (a < b)
        c -= b;

    return (a ^ b) & c;
}

int func3 (unsigned a, unsigned b, int c)
{
    unsigned d = 0;
    unsigned e;

    b++;

    if (a < b)
    {
        int c = e = a - b;

        if (a != 0)
        {
            c -= a & b;
            d = c++ ^ b;
        }
        else
        {
            d++;
            e = c + d ? c : 1;
        }
    }
    d = 1;
    return a < b && (a + b ^ c != 3 || a - b == 0);
}
```

FIG._4A

U.S. Patent May 1, 2001 Sheet 5 of 90 US 6,226,776 B1

```

module funcx (clock,result_func1,result_func2,result_func3,param_func1_a,
    param_func1_b,param_func1_c,param_func2_a,param_func2_b,
    param_func2_c,param_func3_a,param_func3_b,param_func3_c);
    input      clock;
    input [15:0] param_func1_a, param_func1_b, param_func1_c, param_func2_a, param_func2_b,
                param_func2_c, param_func3_a, param_func3_b, param_func3_c;
    output [15:0] result_func1, result_func2, result_func3;
    reg      _5_1;
    reg [15:0] result_func1, result_func2, result_func3,
                _func2_c, _func3_b, _6_func3_d, _6_func3_e,
                _7_func3_c, _1_16, _4_16, _6_16, _7_16;
    reg [16:0] _2_17, _3_17;
    always @(posedge clock)
    begin
        _1_16 = (param_func1_b == 16'd1);
        result_func1 = ((param_func1_a + _1_16) ^ (param_func1_c << 16'd3));
        _func2_c = param_func2_c;
        _2_17 = ( ~ param_func2_a [15], param_func2_a );
        _3_17 = ( ~ param_func2_b [15], param_func2_b );
        if (_2_17 < _3_17)
            _func2_c = (_func2_c - param_func2_b);
        result_func2 = ((param_func2_a ^ param_func2_b) & _func2_c);
        _func3_b = param_func3_b;
        _6_func3_d = 16'd0;
        _func3_b = (_func3_b + 16'd1);
        if ((param_func3_a < _func3_b))
            begin
                _6_func3_e = (param_func3_a - _func3_b);
                _7_func3_c = _6_func3_e;
                if ((param_func3_a != 16'd0))
                    begin
                        _7_func3_c = (_7_func3_c - (param_func3_a & _func3_b));
                        _4_16 = _7_func3_c;
                        _7_func3_c = (_4_16 + 16'd1);
                        _6_func3_d = (_4_16 ^ _func3_b);
                    end
                end
                else
                    begin
                        _6_func3_d = (_6_func3_d + 16'd1);
                        _6_func3_e = ((_7_func3_c + _6_func3_d) ? _7_func3_c : 16'd1);
                    end
            end
        _6_func3_d = 16'd1;
        _6_16 = (param_func3_c != 16'd3);
        _5_1 = (param_func3_a < _func3_b) && (((param_func3_a + _func3_b) ^ _6_16)
            || ((param_func3_a - _func3_b) == 16'd0));
        _7_16 = _5_1;
        result_func3 = _7_16;
    end
endmodule

```

FIG._4B

U.S. Patent May 1, 2001 Sheet 6 of 90 US 6,226,776 B1

```
int sum1 (int n)
{
    int i, sum = 0;
    for (i = 0; i < n; i++)
        sum += i;
    return sum;
}

int sum2 (int array [], int size)
{
    int i, sum = 0;
    for (i = 0; i < size; i++)
        sum += array [i];
    return sum;
}

int main ()
{
    int i;
    int array [10];
    int size = sizeof (array) / sizeof (*array);
    for (i = 0; i < size; i++)
        array [i] = i * 2;
    return sum1 (size) + sum2 (array, size);
}
```

FIG._5A

U.S. Patent

May 1, 2001

Sheet 7 of 90

US 6,226,776 B1

```

`define _5_main_array 4'h1

module RAM32X1 (O, D, WE, A4, A3, A2, A1, A0);
/* synthesis black_box xc_alias="RAM" */
    input      D, WE, A4, A3, A2, A1, A0;
    output     O;
    reg [31:0] d;
    wire [4:0] A = { A4, A3, A2, A1, A0 };
    assign     O = d [A];
    always @ (A or D or WE)
        if (WE)
            d [A] = D;
endmodule

module RAM32x1 (a, d, o, we);
    input [4:0] a;
    input      d, we;
    output     o;
    RAM32X1 RAM32X1 (o, d, we, a [4], a [3], a [2], a [1], a [0]);
endmodule

module RAM32x16 (a, d, o, we);
    input      we;
    input [4:0] a;
    input [15:0] d;
    output [15:0] o;
    RAM32x1 U0 (a, d [0], o [0], we);
    RAM32x1 U1 (a, d [1], o [1], we);
    RAM32x1 U2 (a, d [2], o [2], we);
    RAM32x1 U3 (a, d [3], o [3], we);
    RAM32x1 U4 (a, d [4], o [4], we);
    RAM32x1 U5 (a, d [5], o [5], we);
    RAM32x1 U6 (a, d [6], o [6], we);
    RAM32x1 U7 (a, d [7], o [7], we);
    RAM32x1 U8 (a, d [8], o [8], we);
    RAM32x1 U9 (a, d [9], o [9], we);
    RAM32x1 U10 (a, d [10], o [10], we);
    RAM32x1 U11 (a, d [11], o [11], we);
    RAM32x1 U12 (a, d [12], o [12], we);
    RAM32x1 U13 (a, d [13], o [13], we);
    RAM32x1 U14 (a, d [14], o [14], we);
    RAM32x1 U15 (a, d [15], o [15], we);
endmodule

module RAM11x16 (a, d, o, we);
    input      we;
    input [3:0] a;
    input [15:0] d;
    output [15:0] o;
    RAM32x16 U ({ 1'b0, a }, d, o, we);
endmodule

```

FIG._5B1

U.S. Patent

May 1, 2001

Sheet 8 of 90

US 6,226,776 B1

```

module for1 (clock, reset, result_sum1, result_sum2, result_main,
            param_sum1_n, param_sum2_array, param_sum2_size, run_sum1,
            ready_sum1, run_sum2, ready_sum2, run_main, ready_main);

    input      clock, reset, run_sum1, run_sum2, run_main;
    input [15:0] param_sum1_n, param_sum2_array, param_sum2_size;
    output     ready_sum1, ready_sum2, ready_main;
    output [15:0] result_sum1, result_sum2, result_main;
    reg [15:0]  result_sum1, result_sum2, result_main;
    reg        ready_sum1, ready_sum2, ready_main, we;
    wire [15:0] o, d;
    reg [3:0]   state, a, return_state_sum1, return_state_sum2;
    reg [15:0]  _sum1_n, _2_sum1_i, _2_sum1_sum, _sum2_array, _sum2_size;
    reg [15:0]  _4_sum2_i, _4_sum2_sum, _5_main_i, _5_main_size, _5_16, _8_16;
    reg [16:0]  _1_17, _2_17, _3_17, _4_17, _6_17, _7_17;

    RAM11x16 m (a, d, o, we);
    parameter _sum1 = 1, _sum2 = 5;

    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
            0: begin
                if (run_sum1)
                begin
                    ready_sum1 = 0;
                    return_state_sum1 = 0;
                    _sum1_n = param_sum1_n;
                    state = _sum1;
                end
                else if (run_sum2)
                begin
                    ready_sum2 = 0;
                    return_state_sum2 = 0;
                    _sum2_array = param_sum2_array;
                    _sum2_size = param_sum2_size;
                    state = _sum2;
                end
                else if (run_main)
                begin
                    ready_main = 0;
                    _5_main_size = 16'd10;
                    _5_main_i = 16'd0;
                    state = 9;
                end
            end
        end
    end
end

```

FIG._5B2

U.S. Patent

May 1, 2001

Sheet 9 of 90

US 6,226,776 B1

```

_sum1: begin
    _2_sum1_sum = 16'd0;
    _2_sum1_i = 16'd0;
    state = 2;
end
2: begin
    _1_17 = { ~_2_sum1_i [15], _2_sum1_i };
    _2_17 = { ~_sum1_n [15], _sum1_n };
    state = (_1_17 < _2_17) ? 3 : 4;
end
3: begin
    _2_sum1_sum = (_2_sum1_sum + _2_sum1_i);
    _2_sum1_i = (_2_sum1_i + 16'd1);
    state = 2;
end
4: begin
    result_sum1 = _2_sum1_sum;
    ready_sum1 = 1;
    state = return_state_sum1;
end
sum2: begin
    _4_sum2_sum = 16'd0;
    _4_sum2_i = 16'd0;
    state = 6;
end
6: begin
    _3_17 = { ~_4_sum2_i [15], _4_sum2_i };
    _4_17 = { ~_sum2_size [15], _sum2_size };
    state = (_3_17 < _4_17) ? 7 : 8;
end
7: begin
    a = (_sum2_array + _4_sum2_i);
    state = 12;
end
12: begin
    _5_16 = o;
    _4_sum2_sum = (_4_sum2_sum + _5_16);
    _4_sum2_i = (_4_sum2_i + 16'd1);
    state = 6;
end
8: begin
    result_sum2 = _4_sum2_sum;
    ready_sum2 = 1;
    state = return_state_sum2;
end
9: begin
    _6_17 = { ~_5_main_i [15], _5_main_i };
    _7_17 = { ~_5_main_size [15], _5_main_size };
    state = (_6_17 < _7_17) ? 10 : 11;
end

```

FIG._5B3

U.S. Patent May 1, 2001 Sheet 10 of 90 US 6,226,776 B1

```
10: begin
    _8_16 = (_5_main_i << 1'd1);
    a = ^_5_main_array + _5_main_i);
    d = _8_16;
    we = 1;
    state = 13;
end
13: begin
    we = 0;
    _5_main_i = (_5_main_i + 16'd1);
    state = 9;
end
11: begin
    _sum1_n = _5_main_size;
    return_state_sum1 = 14;
    state = _sum1;
end
14: begin
    _sum2_array = `_5_main_array;
    _sum2_size = _5_main_size;
    return_state_sum2 = 15;
    state = _sum2;
end
15: begin
    result_main = (result_sum1 + result_sum2);
    ready_main = 1;
    state = 0;
end
default: ;
endcase
end
end
endmodule
```

FIG._5B4

U.S. Patent May 1, 2001 Sheet 11 of 90 US 6,226,776 B1

```
#include <stdio.h>

void main ()
{
    int a, b;
    scanf ("%d %d", &a, &b);
    printf ("a + b = %d\n", a + b);
}
```

FIG._6A

```
module Printf (clock, scanf_0_1_line_19_a, scanf_0_2_line_19_b, printf_1_1_line_20, run_main);
    input      clock, run_main;
    input  [15:0] scanf_0_1_line_19_a, scanf_0_2_line_19_b;
    output [15:0] printf_1_1_line_20;
    reg   [15:0] printf_1_1_line_20, _3_main_a, _3_main_b;

    always @(posedge clock)
    begin
        if (run_main)
        begin
            _3_main_a = scanf_0_1_line_19_a;
            _3_main_b = scanf_0_2_line_19_b;
            printf_1_1_line_20 = (_3_main_a + _3_main_b);

            // User Verilog code
            $write ("a + b = %d\n", printf_1_1_line_20);
            // End of user Verilog code
        end
    end
endmodule
```

FIG._6B

U.S. Patent May 1, 2001 Sheet 12 of 90 US 6,226,776 B1

```
int func (a, b, c, d)
{
    return (a / b + c / d);
}
```

FIG._7A

```
module udivmod16 (a, b, div, mod);
    input [15:0] a, b;
    output [15:0] div, mod;
    reg [15:0] div, mod, pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7, pa8, pa9, pa10, pa11, pa12,
                pa13, pa14, pa15, pa16, shb0;
    reg [16:0] shb1;
    reg [17:0] shb2;
    reg [18:0] shb3;
    reg [19:0] shb4;
    reg [20:0] shb5;
    reg [21:0] shb6;
    reg [22:0] shb7;
    reg [23:0] shb8;
    reg [24:0] shb9;
    reg [25:0] shb10;
    reg [26:0] shb11;
    reg [27:0] shb12;
    reg [28:0] shb13;
    reg [29:0] shb14;
    reg [30:0] shb15;

    always @(a or b)
    begin
        pa0 = a;

        shb15 = b << 15;
        div [15] = pa0 >= shb15;
        pa1 = div [15] ? pa0 - shb15 : pa0;

        shb14 = b << 14;
        div [14] = pa1 >= shb14;
        pa2 = div [14] ? pa1 - shb14 : pa1;

        shb13 = b << 13;
        div [13] = pa2 >= shb13;
        pa3 = div [13] ? pa2 - shb13 : pa2;
```

FIG._7B1

U.S. Patent May 1, 2001 Sheet 13 of 90 US 6,226,776 B1

```

shb12 = b << 12;
div [12] = pa3 >= shb12;
pa4 = div [12] ? pa3 - shb12 : pa3;
shb11 = b << 11;
div [11] = pa4 >= shb11;
pa5 = div [11] ? pa4 - shb11 : pa4;
shb10 = b << 10;
div [10] = pa5 >= shb10;
pa6 = div [10] ? pa5 - shb10 : pa5;
shb9 = b << 9;
div [ 9] = pa6 >= shb9;
pa7 = div [ 9] ? pa6 - shb9 : pa6;
shb8 = b << 8;
div [ 8] = pa7 >= shb8;
pa8 = div [ 8] ? pa7 - shb8 : pa7;
shb7 = b << 7;
div [ 7] = pa8 >= shb7;
pa9 = div [ 7] ? pa8 - shb7 : pa8;
shb6 = b << 6;
div [ 6] = pa9 >= shb6;
pa10 = div [ 6] ? pa9 - shb6 : pa9;
shb5 = b << 5;
div [ 5] = pa10 >= shb5;
pa11 = div [ 5] ? pa10 - shb5 : pa10;
shb4 = b << 4;
div [ 4] = pa11 >= shb4;
pa12 = div [ 4] ? pa11 - shb4 : pa11;
shb3 = b << 3;
div [ 3] = pa12 >= shb3;
pa13 = div [ 3] ? pa12 - shb3 : pa12;
shb2 = b << 2;
div [ 2] = pa13 >= shb2;
pa14 = div [ 2] ? pa13 - shb2 : pa13;
shb1 = b << 1;
div [ 1] = pa14 >= shb1;
pa15 = div [ 1] ? pa14 - shb1 : pa14;
shb0 = b << 0;
div [ 0] = pa15 >= shb0;
pa16 = div [ 0] ? pa15 - shb0 : pa15;
mod = pa16;
end
endmodule

```

FIG._7B2

U.S. Patent May 1, 2001 Sheet 14 of 90 US 6,226,776 B1

```

module sdivmod16 (a, b, div, mod);

input [15:0] a, b;
output [15:0] div, mod;

wire [15:0] ua = a [15] ? - a : a;
wire [15:0] ub = b [15] ? - b : b;
wire [15:0] udiv, umod;

udivmod16 udivmod (ua, ub, udiv, umod);

wire [15:0] div = (a [15] != b [15]) ? - udiv : udiv;
wire [15:0] mod = a [15] ? - umod : umod;

endmodule

```

```

module DivEx (clock,reset,result_func,param_func_a,param_func_b,
param_func_c,param_func_d,run_func,ready_func);

input      clock, reset, run_func;
input [15:0] param_func_a, param_func_b, param_func_c, param_func_d;
output     ready_func;
output [15:0] result_func;

reg [15:0] result_func, divmod_a, divmod_b, _1_16, _2_16;
reg        ready_func;
reg [1:0]   state;
wire [15:0] div_result;

sdivmod16 sdivmod (divmod_a, divmod_b, div_result, );

always @(posedge clock)
begin
    if (reset)
        state = 0;
    else
        begin
            case (state)
                0: begin
                    if (run_func)
                        begin
                            ready_func = 0;
                            divmod_a = param_func_a;
                            divmod_b = param_func_b;
                            state = 1;
                        end
                end
            end
        end
end

```

FIG._7B3

U.S. Patent May 1, 2001 Sheet 15 of 90 US 6,226,776 B1

```
1: begin
    _1_16 = div_result;
    divmod_a = param_func_c;
    divmod_b = param_func_d;
    state = 2;
end

2: begin
    _2_16 = div_result;
    result_func = (_1_16 + _2_16);
    ready_func = 1;
    state = 0;
end

default: ;

endcase
end
end
```

endmodule

FIG._7B4

U.S. Patent May 1, 2001 Sheet 16 of 90 US 6,226,776 B1

```
int data_in;
int out1;
int out2;
int data_out;

void pipeline_stage_1 ()
{
    out1 = data_in + 1;
}

void pipeline_stage_2 ()
{
    out2 = out1 + out1;
}

void pipeline_stage_3 ()
{
    data_out = out2 ^ 1234;
}

#ifndef __SYNTRY__
#include <stdio.h>

void main ()
{
    for (;;)
    {
        scanf ("%d", & data_in);

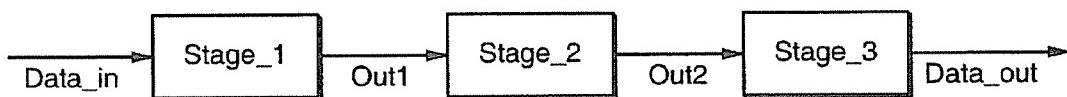
        pipeline_stage_1 ();
        pipeline_stage_2 ();
        pipeline_stage_3 ();

        printf ("%d", data_out);
    }
}
#endif
```

FIG._8A

U.S. Patent May 1, 2001 Sheet 17 of 90 US 6,226,776 B1

```
module Pipeline (clock, _data_in, _data_out, run_pipeline_stage_1, run_pipeline_stage_2,
run_pipeline_stage_3);
    input      clock, run_pipeline_stage_1, run_pipeline_stage_2, run_pipeline_stage_3;
    input [15:0] _data_in;
    output [15:0] _data_out;
    reg   [15:0] _data_out, _out1, _out2;
    always @(posedge clock)
    begin
        if (run_pipeline_stage_1)
            _out1 = (_data_in + 16'd1);
    end
    always @(posedge clock)
    begin
        if (run_pipeline_stage_2)
            _out2 = (_out1 + _out1);
    end
    always @(posedge clock)
    begin
        if (run_pipeline_stage_3)
            _data_out = (_out2 ^ 16'd1234);
    end
endmodule
```

FIG._8B**FIG._8C**

U.S. Patent

May 1, 2001

Sheet 18 of 90

US 6,226,776 B1

```
int func (float a, double b, long double c, int i)
{
    return - a + b * c - 16.5 / c;
}
```

FIG._9A

```
module Float (clock,reset,zero_substractor_float_operand,
    zero_substractor_float_run,zero_substractor_float_ready,zero_substractor_float_result,
    convertor_from_float_to_long_double_operand,convertor_from_float_to_long_double_run,
    convertor_from_float_to_long_double_ready,convertor_from_float_to_long_double_result,
    convertor_from_double_to_long_double_operand,convertor_from_double_to_long_double_run,
    convertor_from_double_to_long_double_ready,convertor_from_double_to_long_double_result,
    multiplier_long_double_a,multiplier_long_double_b,multiplier_long_double_run,
    multiplier_long_double_ready,multiplier_long_double_result,adder_long_double_a,
    adder_long_double_b,adder_long_double_run,adder_long_double_ready,adder_long_double_result,
    divider_long_double_a,divider_long_double_b,divider_long_double_run,
    divider_long_double_ready,divider_long_double_result,substractor_long_double_a,
    substractor_long_double_b,substractor_long_double_run,substractor_long_double_ready,
    substractor_long_double_result,convertor_from_long_double_to_signed_16_operand,
    convertor_from_long_double_to_signed_16_run,convertor_from_long_double_to_signed_16_ready,con
    vertor_from_long_double_to_signed_16_result,result_func,param_func_a,param_func_b,
    param_func_c,param_func_i,run_func,ready_func);

input      clock, reset, zero_substractor_float_ready, convertor_from_float_to_long_double_ready,
           convertor_from_double_to_long_double_ready, multiplier_long_double_ready,
           adder_long_double_ready, divider_long_double_ready, substractor_long_double_ready,
           convertor_from_long_double_to_signed_16_ready, run_func;
input [15:0] convertor_from_long_double_to_signed_16_result, param_func_i;
input [31:0] zero_substractor_float_result, param_func_a;
input [63:0] param_func_b;
input [79:0] convertor_from_float_to_long_double_result, param_func_c,
           convertor_from_double_to_long_double_result, multiplier_long_double_ready,
           adder_long_double_result, divider_long_double_result, substractor_long_double_result;
output     zero_substractor_float_run, convertor_from_float_to_long_double_run,
           convertor_from_double_to_long_double_run, multiplier_long_double_run,
           adder_long_double_run, divider_long_double_run, substractor_long_double_run,
           convertor_from_long_double_to_signed_16_run, ready_func;
output [15:0] result_func;
output [31:0] zero_substractor_float_operand, convertor_from_float_to_long_double_operand;
output [63:0] convertor_from_double_to_long_double_operand;
output [79:0] multiplier_long_double_a, multiplier_long_double_b, adder_long_double_a,
           adder_long_double_b, divider_long_double_a, divider_long_double_b,
           substractor_long_double_a, substractor_long_double_b,
           convertor_from_long_double_to_signed_16_operand;
```

FIG._9B1

U.S. Patent May 1, 2001 Sheet 19 of 90 US 6,226,776 B1

```

reg      zero_substractor_float_run, convertor_from_float_to_long_double_run,
convertor_from_double_to_long_double_run, multiplier_long_double_ran,
adder_long_double_run, divider_long_double_run, subtractor_long_double_run,
convertor_from_long_double_to_signed_16_run, ready_func;
reg [4:0] state;
reg [15:0] result_func, _8_16;
reg [31:0] zero_substractor_float_operand, convertor_from_float_to_long_double_operand, _1_32;
reg [63:0] convertor_from_double_to_long_double_operand;
reg [79:0] multiplier_long_double_a, multiplier_long_double_b, adder_long_double_a,
adder_long_double_b, divider_long_double_a, divider_long_double_b,
substractor_long_double_a, substractor_long_double_b,
convertor_from_long_double_to_signed_16_operand, _2_80, _3_80, _4_80, _5_80,
_6_80, _7_80;

always @(posedge clock)
begin
  if (reset)
    state = 0;
  else
    begin
      case (state)
        0: begin
          if (run_func)
            begin
              ready_func = 0;
              zero_substractor_float_operand = param_func_a;
              zero_substractor_float_run = 1;
              state = 1;
            end
        end
        1: begin
          zero_substractor_float_run = 0;
          _1_32 = zero_substractor_float_result;
          if (zero_substractor_float_ready)
            state = 2;
        end
        2: begin
          convertor_from_float_to_long_double_operand = _1_32;
          convertor_from_float_to_long_double_run = 1;
          state = 3;
        end
        3: begin
          convertor_from_float_to_long_double_run = 0;
          _2_80 = convertor_from_float_to_long_double_result;
          if (convertor_from_float_to_long_double_ready)
            state = 4;
        end
      endcase
    end
  end
end

```

FIG._ 9B2

U.S. Patent May 1, 2001 Sheet 20 of 90 US 6,226,776 B1

```

4: begin
    convertor_from_double_to_long_double_operand = param_func_b;
    convertor_from_double_to_long_double_run = 1;
    state = 5;
end
5: begin
    convertor_from_double_to_long_double_run = 0;
    _3_80 = convertor_from_double_to_long_double_result;
    if (convertor_from_double_to_long_double_ready)
        state = 6;
end
6: begin
    multiplier_long_double_a = _3_80;
    multiplier_long_double_b = param_func_c;
    multiplier_long_double_run = 1;
    state = 7;
end
7: begin
    multiplier_long_double_run = 0;
    _4_80 = multiplier_long_double_result;
    if (multiplier_long_double_ready)
        state = 8;
end
8: begin
    adder_long_double_a = _2_80;
    adder_long_double_b = _4_80;
    adder_long_double_run = 1;
    state = 9;
end
9: begin
    adder_long_double_run = 0;
    _5_80 = adder_long_double_result;
    if (adder_long_double_ready)
        state = 10;
end
10: begin
    divider_long_double_a = 80'h00000000000803040;
    divider_long_double_b = param_func_c;
    divider_long_double_run = 1;
    state = 11;
end
11: begin
    divider_long_double_run = 0;
    _6_80 = divider_long_double_result;
    if (divider_long_double_ready)
        state = 12;
end

```

FIG._9B3

U.S. Patent May 1, 2001 Sheet 21 of 90 US 6,226,776 B1

```
12: begin
    substractor_long_double_a = _5_80;
    substractor_long_double_b = _6_80;
    substractor_long_double_run = 1;
    state = 13;
end
13: begin
    substractor_long_double_run = 0;
    _7_80 = substractor_long_double_result;
    if (substractor_long_double_ready)
        state = 14;
end
14: begin
    convertor_from_long_double_to_signed_16_operand = _7_80;
    convertor_from_long_double_to_signed_16_run = 1;
    state = 15;
end
15: begin
    convertor_from_long_double_to_signed_16_run = 0;
    _8_16 = convertor_from_long_double_to_signed_16_result;
    if (convertor_from_long_double_to_signed_16_ready)
        state = 16;
end
16: begin
    result_func = _8_16;
    ready_func = 1;
    state = 0;
end
default: ;
endcase
end
end
```

endmodule

FIG._9B4

U.S. Patent May 1, 2001 Sheet 22 of 90 US 6,226,776 B1

```
typedef struct
{
    int real;
    int image;
}
Complex;
```

```
Complex add (Complex a, Complex b)
{
    Complex c;
    c.real = a.real + b.real;
    c.image = a.image + b.image;
    return c;
}
```

```
Complex x = { 1, 2 }, y = { 3, 4 }, z;
```

```
void main ()
{
    z = add (x, add (x, y));
}
```

FIG._10A

U.S. Patent May 1, 2001 Sheet 23 of 90 US 6,226,776 B1

```

`define _add_a 4'h1
`define _add_b 4'h3
`define _3_add_c 4'h5
`define _x      4'h7
`define _y      4'h9
`define _z      4'hB

module Complex (clock, reset, a, d, o, we, result_add, param_add_a,
                param_add_b, run_add, ready_add, run_main, ready_main);
    input      clock, reset, run_add, run_main;
    input [15:0] o;
    input [31:0] param_add_a, param_add_b;
    output     we, ready_add, ready_main;
    output [ 3:0] a;
    output [15:0] d;
    output [31:0] result_add;

    reg      we, ready_add, ready_main;
    reg [ 3:0] a;
    reg [ 4:0] state, return_state_add;
    reg [15:0] d, _1_16, _2_16, _3_16, _4_16, _5_16, _6_16;
    reg [31:0] result_add, _7_32, _8_32, _9_32, _10_32;

    parameter _add = 1;

    always @(posedge clock)
    begin
        if (reset)
            begin
                we = 0;
                state = 0;
            end
        else
            begin
                case (state)
                    0: begin
                        if (run_add)
                            begin
                                ready_add = 0;
                                return_state_add = 0;
                                a = `_add_a;
                                d = param_add_a [31:16];
                                we = 1;
                                state = 2;
                            end
                        else if (run_main)
                            begin
                                ready_main = 0;
                                a = `_x;
                                state = 14;
                            end
                end
            end
    end

```

FIG._ 10B1

U.S. Patent May 1, 2001 Sheet 24 of 90 US 6,226,776 B1

```

2: begin
    we = 0;
    a = `_add_a + 4'd1;
    d = param_add_a [15:0];
    we = 1;
    state = 3;
end

3: begin
    we = 0;
    a = `_add_b;
    d = param_add_b [31:16];
    we = 1;
    state = 4;
end

4: begin
    we = 0;
    a = `_add_b + 4'd1;
    d = param_add_b [15:0];
    we = 1;
    state = 5;
end

5: begin
    we = 0;
    state = _add;
end

_add: begin
    a = `_add_a;
    state = 6;
end

6: begin
    _1_16 = o;
    a = `_add_b;
    state = 7;
end

7: begin
    _2_16 = o;
    _3_16 = (_1_16 + _2_16);
    a = `_3_add_c;
    d = _3_16;
    we = 1;
    state = 8;
end

8: begin
    we = 0;
    a = (_add_a + 16'd1);
    state = 9;
end

```

FIG._ 10B2

U.S. Patent

May 1, 2001

Sheet 25 of 90

US 6,226,776 B1

```

9: begin
    _4_16 = o;
    a = `_add_b + 16'd1);
    state = 10;
end

10: begin
    _5_16 = o;
    _6_16 = (_4_16 + _5_16);
    a = `_3_add_c + 16'd1);
    d = _6_16;
    we = 1;
    state = 11;
end

11: begin
    we = 0;
    a = `_3_add_c;
    state = 12;
end

12: begin
    _7_32 [31:16] = o;
    a = `_3_add_c + 4'd1;
    state = 13;
end

13: begin
    _7_32 [15:0] = o;
    result_addr = _7_32;
    ready_addr = 1;
    state = return_state_addr;
end

14: begin
    _8_32 [31:16] = o;
    a = `_x + 4'd1;
    state = 15;
end

15: begin
    _8_32 [15:0] = o;
    a = `_add_a;
    d = _8_32 [31:16];
    we = 1;
    state = 16;
end

16: begin
    we = 0;
    a = `_add_a + 4'd1;
    d = _8_32 [15:0];
    we = 1;
    state = 17;
end

```

FIG._ 10B3

U.S. Patent

May 1, 2001

Sheet 26 of 90

US 6,226,776 B1

```

17: begin
    we = 0;
    a = `_x;
    state = 18;
end

18: begin
    _9_32 [31:16] = o;
    a = `_x + 4'd1;
    state = 19;
end

19: begin
    _9_32 [15:0] = o;
    a = `_add_a;
    d = _9_32 [31:16];
    we = 1;
    state = 20;
end

20: begin
    we = 0;
    a = `_add_a + 4'd1;
    d = _9_32 [15:0];
    we = 1;
    state = 21;
end

21: begin
    we = 0;
    a = `_y;
    state = 22;
end

22: begin
    _10_32 [31:16] = o;
    a = `_y + 4'd1;
    state = 23;
end

23: begin
    _10_32 [15:0] = o;
    a = `_add_b;
    d = _10_32 [31:16];
    we = 1;
    state = 24;
end

24: begin
    we = 0;
    a = `_add_b + 4'd1;
    d = _10_32 [15:0];
    we = 1;
    state = 25;
end

```

FIG._ 10B4

U.S. Patent

May 1, 2001

Sheet 27 of 90

US 6,226,776 B1

```

25: begin
    we = 0;
    return_state_add = 26;
    state = _add;
end

26: begin
    a = `_add_b;
    d = result_add [31:16];
    we = 1;
    state = 27;
end

27: begin
    we = 0;
    a = `_add_b + 4'd1;
    d = result_add [15:0];
    we = 1;
    state = 28;
end

28: begin
    we = 0;
    return_state_add = 29;
    state = _add;
end

29: begin
    a = `_z;
    d = result_add [31:16];
    we = 1;
    state = 30;
end

30: begin
    we = 0;
    a = `_z + 4'd1;
    d = result_add [15:0];
    we = 1;
    state = 31;
end

31: begin
    we = 0;
    ready_main = 1;
    state = 0;
end

default: ;
endcase
end
end

endmodule

```

FIG._ 10B5

U.S. Patent

May 1, 2001

Sheet 28 of 90

US 6,226,776 B1

```

#ifndef __SYNETRY__
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#else
#define NULL ((NODE *) 0)
#define assert(a)
#endif
#define MAX_NODES 10
struct _Node
{
    struct _Node * pLeft;
    struct _Node * pRight;
    int      nKey;
    int      nValue;
};
typedef struct _Node NODE;
static NODE Nodes [MAX_NODES];
static int nNodes;
NODE * pTree;
void Initialize (void)
{
    nNodes = 0;
    pTree = NULL;
}
static NODE * NewNode (int nKey, int nValue)
{
    assert (nNodes >= 0 && nNodes <= MAX_NODES);
    if (nNodes == MAX_NODES)
        return NULL;
    Nodes [nNodes].pLeft = NULL;
    Nodes [nNodes].pRight = NULL;
    Nodes [nNodes].nKey  = nKey;
    Nodes [nNodes].nValue = nValue;
    return & Nodes [nNodes++];
}

NODE * FindNode (int nKey)
{
    NODE * p;
    p = pTree;
    while (p != NULL)
    {
        if (p -> nKey < nKey)
            p = p -> pLeft;
        else if (p -> nKey > nKey)
            p = p -> pRight;
        else
            return p;
    }
    return NULL;
}
int FindValue (int nKey)
{
    NODE * p;
    return (p = FindNode (nKey)) == NULL ? -1 : p -> nValue;
}

```

FIG._ 11A1

U.S. Patent May 1, 2001 Sheet 29 of 90 US 6,226,776 B1

```

NODE * FindOrAddNode (nKey, nValue)
int nKey;
int nValue;
{ NODE ** p;
  p = & pTree;
  while (*p != NULL)
  {   if ((*p) -> nKey < nKey)
      p = & (*p) -> pLeft;
    else if ((*p) -> nKey > nKey)
      p = & (*p) -> pRight;
    else
      return (*p);
  }
  return *p = NewNode (nKey, nValue);
}
int FindOrAddValue (nKey, nValue)
int nKey;
int nValue;
{ NODE * p = FindOrAddNode (nKey, nValue);
  return p == NULL ? -1 : p -> nValue;
}
#ifndef __SYNETRY__
void PrintTree (int nLevel, NODE * p)
{ if (p == NULL)
  return;
  PrintTree (nLevel + 1, p -> pLeft);
  printf ("%*s%d [%d]\n", nLevel * 4, "", p -> nKey, p -> nValue);
  PrintTree (nLevel + 1, p -> pRight);
}

void main ()
{ int anKeys [20]
  = { 12, 4, 7, 2, 18, 24, 0, 5, 14, 1, 43, 6, 19, 21, 26, 11, 37, 8, 4, 9};
  int i;

  Initialize ();
  for (i = 0; i < 20; i++)
    FindOrAddValue (anKeys [i], i);
  for (i = 0; i < 20; i++)
    printf ("%d: %d\n", i, FindValue (i));
  printf ("\nTree:\n");
  PrintTree (0, pTree);
}
#endif

```

FIG._ 11A2

U.S. Patent May 1, 2001 Sheet 30 of 90 US 6,226,776 B1

```

`define _Nodes 6'h1
`define _pTree 6'h29

module Tree ( clock, reset, a, d, o, we, result_FindNode, result_FindValue, result_FindOrAddNode,
              result_FindOrAddValue, param_FindNode_nKey, param_FindValue_nKey,
              param_FindOrAddNode_nKey, param_FindOrAddNode_nValue,
              param_FindOrAddValue_nKey, param_FindOrAddValue_nValue,
              run_Initialize, ready_Initialize, run_FindNode, ready_FindNode,
              run_FindValue, ready_FindValue, run_FindOrAddNode,
              ready_FindOrAddNode, run_FindOrAddValue, ready_FindOrAddValue );
    input   clock, reset, run_Initialize, run_FindNode, run_FindValue, run_FindOrAddNode,
            run_FindOrAddValue;
    input [15:0] o, param_FindNode_nKey, param_FindValue_nKey, param_FindOrAddNode_nKey,
                param_FindOrAddNode_nValue, param_FindOrAddValue_nKey,
                param_FindOrAddValue_nValue;
    output  we, ready_Initialize, ready_FindNode, ready_FindValue, ready_FindOrAddNode,
            ready_FindOrAddValue;
    output [ 5:0] a, result_FindNode, result_FindOrAddNode, d, result_FindValue, result_FindOrAddValue;
    reg      we, ready_Initialize, ready_FindNode, ready_FindValue, ready_FindOrAddNode,
            ready_FindOrAddValue;
    reg [ 5:0] a, result_FindNode, result_FindOrAddNode, d, result_FindValue, result_FindOrAddValue,
                state, result_NewNode, _7_FindNode_p, _10_FindValue_p, _13_FindOrAddNode_p,
                _17_FindOrAddValue_p, return_state_NewNode, return_state_FindNode,
                return_state_FindOrAddNode, _1_6, _2_6, _3_6, _4_6, _5_6, _6_6, _7_6, _8_6, _9_6,
                _10_6, _12_6, _13_6, _14_6, _18_6, _22_6, _23_6, _24_6, _27_6, _28_6, _29_6,
                _33_6, _34_6, _38_6, _39_6, _40_6;
    reg [15:0] _nNodes, _NewNode_nKey, _NewNode_nValue, _FindNode_nKey,
                _FindOrAddNode_nKey, _FindOrAddNode_nValue, _30_16, _35_16, _11_16, _15_16,
                _19_16, _25_16, _26_16, _41_16, _42_16;
    reg [16:0] _16_17, _17_17, _20_17, _21_17, _31_17, _32_17, _36_17, _37_17;

    parameter _NewNode      = 1,
               _FindNode     = 5,
               _FindOrAddNode = 14;

    always @(posedge clock)
    begin
        if (reset)
            begin
                we = 0;
                state = 0;
            end
    end

```

FIG._ 11B1

U.S. Patent May 1, 2001 Sheet 31 of 90 US 6,226,776 B1

```

else
begin
  case (state)
  0: begin
    if (run_Initialize)
    begin
      ready_Initialize = 0;
      _nNodes = 16'd0;
      _1_6 = 16'd0;
      a = `pTree;
      d [15:10] = _1_6;
      we = 1;
      state = 23;
    end
  else if (run_FindNode)
  begin
    ready_FindNode = 0;
    return_state_FindNode = 0;
    _FindNode_nKey = param_FindNode_nKey;
    state = _FindNode;
  end
  else if (run_FindValue)
  begin
    ready_FindValue = 0;
    _FindNode_nKey = param_FindValue_nKey;
    return_state_FindNode = 33;
    state = _FindNode;
  end
  else if (run_FindOrAddNode)
  begin
    ready_FindOrAddNode = 0;
    return_state_FindOrAddNode = 0;
    _FindOrAddNode_nKey = param_FindOrAddNode_nKey;
    _FindOrAddNode_nValue = param_FindOrAddNode_nValue;
    state = _FindOrAddNode;
  end
  else if (run_FindOrAddValue)
  begin
    ready_FindOrAddValue = 0;
    _FindOrAddNode_nKey = param_FindOrAddValue_nKey;
    _FindOrAddNode_nValue = param_FindOrAddValue_nValue;
    return_state_FindOrAddNode = 48;
    state = _FindOrAddNode;
  end
end
23: begin
  we = 0;
  ready_Initialize = 1;
  state = 0;
end
_NewNode: state = (_nNodes == 16'd10) ? 2 : 3;

```

FIG._ 11B2

U.S. Patent

May 1, 2001

Sheet 32 of 90

US 6,226,776 B1

```

2: begin
    _2_6 = 16'd0;
    result_NewNode = _2_6;
    state = 4;
end

3: begin
    _3_6 = 16'd0;
    _4_6 = (_nNodes << 2'd2);
    _5_6 = _3_6;
    a = (_Nodes + _4_6);
    d [15:10] = _5_6;
    we = 1;
    state = 24;
end

24: begin
    we = 0;
    _6_6 = 16'd0;
    _7_6 = (_nNodes << 2'd2);
    _8_6 = _6_6;
    a = ((_Nodes + _7_6) + 6'd1);
    d [15:10] = _8_6;
    we = 1;
    state = 25;
end

25: begin
    we = 0;
    _9_6 = (_nNodes << 2'd2);
    a = ((_Nodes + _9_6) + 6'd2);
    d = _NewNode_nKey;
    we = 1;
    state = 26;
end

26: begin
    we = 0;
    _10_6 = (_nNodes << 2'd2);
    a = ((_Nodes + _10_6) + 6'd3);
    d = _NewNode_nValue;
    we = 1;
    state = 27;
end

27: begin
    we = 0;
    _11_16 = _nNodes;
    _nNodes = (_11_16 + 16'd1);
    _12_6 = (_11_16 << 2'd2);
    result_NewNode = (_Nodes + _12_6);
    state = 4;
end

```

FIG._ 11B3

U.S. Patent May 1, 2001 Sheet 33 of 90 US 6,226,776 B1

```

4: state = return_state_NewNode;

FindNode: begin
    a = `_pTree;
    state = 28;
end

28: begin
    _13_6 = o [15:10];
    _7_FindNode_p = _13_6;
    state = 6;
end

6: begin
    _14_6 = 16'd0;
    state = (_7_FindNode_p != _14_6) ? 7 : 12;
end

7: begin
    a = (_7_FindNode_p + 6'd2);
    state = 29;
end

29: begin
    _15_16 = o;
    _16_17 = { ~_15_16 [15], _15_16 };
    _17_17 = { ~_FindNode_nKey [15], _FindNode_nKey };
    state = (_16_17 < _17_17) ? 8 : 9;
end

8: begin
    a = _7_FindNode_p;
    state = 30;
end

30: begin
    _18_6 = o [15:10];
    _7_FindNode_p = _18_6;
    state = 6;
end

9: begin
    a = (_7_FindNode_p + 6'd2);
    state = 31;
end

31: begin
    _19_16 = o;
    _20_17 = { ~_19_16 [15], _19_16 };
    _21_17 = { ~_FindNode_nKey [15], _FindNode_nKey };
    state = (_20_17 > _21_17) ? 10 : 11;
end

10: begin
    a = (_7_FindNode_p + 6'd1);
    state = 32;
end

```

FIG._ 11B4

U.S. Patent May 1, 2001 Sheet 34 of 90 US 6,226,776 B1

```

32: begin
    _22_6 = o [15:10];
    _7_FindNode_p = _22_6;
    state = 6;
end

11: begin
    result_FindNode = _7_FindNode_p;
    state = 13;
end

12: begin
    _23_6 = 16'd0;
    result_FindNode = _23_6;
    state = 13;
end

13: begin
    ready_FindNode = 1;
    state = return_state_FindNode;
end

33: begin
    _10_FindValue_p = result_FindNode;
    _24_6 = 16'd0;
    state = (result_FindNode == _24_6) ? 34 : 35;
end

34: begin
    _25_16 = -16'd1;
    state = 36;
end

35: begin
    a = (_10_FindValue_p + 6'd3);
    state = 37;
end

37: begin
    _26_16 = o;
    _25_16 = _26_16;
    state = 36;
end

36: begin
    result_FindValue = _25_16;
    ready_FindValue = 1;
    state = 0;
end

```

FIG._ 11B5

U.S. Patent May 1, 2001 Sheet 35 of 90 US 6,226,776 B1

```

_FindOrAddNode: begin
    _13_FindOrAddNode_p = `_pTree;
    state = 15;
end

15: begin
    a = _13_FindOrAddNode_p;
    state = 38;
end

38: begin
    _27_6 = o [15:10];
    _28_6 = 16'd0;
    state = (_27_6 != _28_6) ? 16 : 21;
end

16: begin
    a = _13_FindOrAddNode_p;
    state = 39;
end

39: begin
    _29_6 = o [15:10];
    a = (_29_6 + 6'd2);
    state = 40;
end

40: begin
    _30_16 = o;
    _31_17 = { ~_30_16 [15], _30_16 };
    _32_17 = { ~_FindOrAddNode_nKey [15], _FindOrAddNode_nKey };
    state = (_31_17 < _32_17) ? 17 : 18;
end

17: begin
    a = _13_FindOrAddNode_p;
    state = 41;
end

41: begin
    _33_6 = o [15:10];
    _13_FindOrAddNode_p = _33_6;
    state = 15;
end

18: begin
    a = _13_FindOrAddNode_p;
    state = 42;
end

42: begin
    _34_6 = o [15:10];
    a = (_34_6 + 6'd2);
    state = 43;
end

```

FIG._ 11B6

U.S. Patent May 1, 2001 Sheet 36 of 90 US 6,226,776 B1

```

43: begin
    _35_16 = o;
    _36_17 = { ~_35_16 [15], _35_16 };
    _37_17 = { ~_FindOrAddNode_nKey [15], _FindOrAddNode_nKey };
    state = (_36_17 > _37_17) ? 19 : 20;
end

19: begin
    a = _13_FindOrAddNode_p;
    state = 44;
end

44: begin
    _38_6 = o [15:10];
    _13_FindOrAddNode_p = (_38_6 + 6'd1);
    state = 15;
end

20: begin
    a = _13_FindOrAddNode_p;
    state = 45;
end

45: begin
    _39_6 = o [15:10];
    result_FindOrAddNode = _39_6;
    state = 22;
end

21: begin
    _NewNode_nKey = _FindOrAddNode_nKey;
    _NewNode_nValue = _FindOrAddNode_nValue;
    return_state_NewNode = 46;
    state = _NewNode;
end

46: begin
    a = _13_FindOrAddNode_p;
    d [15:10] = result_NewNode;
    we = 1;
    state = 47;
end

47: begin
    we = 0;
    result_FindOrAddNode = result_NewNode;
    state = 22;
end

```

FIG._ 11B7

U.S. Patent May 1, 2001 Sheet 37 of 90 US 6,226,776 B1

```
22: begin
    ready_FindOrAddNode = 1;
    state = return_state_FindOrAddNode;
end

48: begin
    _17_FindOrAddValue_p = result_FindOrAddNode;
    _40_6 = 16'd0;
    state = (_17_FindOrAddValue_p == _40_6) ? 49 : 50;
end

49: begin
    _41_16 = -16'd1;
    state = 51;
end

50: begin
    a = (_17_FindOrAddValue_p + 6'd3);
    state = 52;
end

52: begin
    _42_16 = 0;
    _41_16 = _42_16;
    state = 51;
end

51: begin
    result_FindOrAddValue = _41_16;
    ready_FindOrAddValue = 1;
    state = 0;
end

default: ;
endcase
end
end

endmodule
```

FIG._ 11B8

U.S. Patent May 1, 2001 Sheet 38 of 90 US 6,226,776 B1

```
#include <stdio.h>
int event_occurred = 0;
#pragma synchronous on_event

void on_event ()
{
    event_occurred = 1;
}

void main ()
{
    int a, b, c;
    scanf ("%d %d", &a, &b);
    c = event_occurred ? a + b : a - b;
    printf ("%d\n", c);
}
```

FIG._ 12A

```
module Event_sy (clock,reset,scanf_0_1_line_16_a,scanf_0_2_line_16_b,printf_1_1_line_20_c,
                run_on_event,run_main);
    input      clock, reset, run_on_event, run_main;
    input [15:0] scanf_0_1_line_16_a, scanf_0_2_line_16_b;
    output [15:0] printf_1_1_line_20_c;
    reg   [15:0] printf_1_1_line_20_c, _event_occurred, _108_main_a, _108_main_b, _108_main_c;

    always @(posedge clock)
    begin
        if (run_on_event)
            _event_occurred = 16'd1;
        if (reset)
            _event_occurred = 16'd0;
    end

    always @(posedge clock)
    begin
        if (run_main)
        begin
            _108_main_a = scanf_0_1_line_16_a;
            _108_main_b = scanf_0_2_line_16_b;
            _108_main_c = (_event_occurred ? (_108_main_a + _108_main_b) :
                           (_108_main_a - _108_main_b));
            printf_1_1_line_20_c = _108_main_c;
        end
    end
endmodule
```

FIG._ 12B

U.S. Patent May 1, 2001 Sheet 39 of 90 US 6,226,776 B1

```
#include <stdio.h>

int event_occurred = 0;
#pragma asynchronous on_event
void on_event()
{
    event_occurred = 1;
}

void main()
{
    int a, b, c;
    scanf ("%d %d", &a, &b);
    c = event_occurred ? a + b : a - b;
    printf ("%d\n", c);
}
```

FIG._ 13A

```
module Event_as(clock,reset,scanf_0_1_line_16_a,scanf_0_2_line_16_b,printf_1_1_line_20_c,
               run_on_event,run_main);
    input      clock, reset, run_on_event, run_main;
    input [15:0] scanf_0_1_line_16_a, scanf_0_2_line_16_b;
    output [15:0] printf_1_1_line_20_c;
    reg   [15:0] printf_1_1_line_20_c, _event_occurred, _108_main_a, _108_main_b, _108_main_c;

    always @(posedge run_on_event)
    begin
        if (reset)
            _event_occurred = 16'd0;
        else
            _event_occurred = 16'd1;
    end

    always @(posedge clock)
    begin
        if (run_main)
        begin
            _108_main_a = scanf_0_1_line_16_a;
            _108_main_b = scanf_0_2_line_16_b;
            _108_main_c = (_event_occurred ? (_108_main_a + _108_main_b) :
                           (_108_main_a - _108_main_b));
            printf_1_1_line_20_c = _108_main_c;
        end
    end
end
```

endmodule

FIG._ 13B

U.S. Patent May 1, 2001 Sheet 40 of 90 US 6,226,776 B1

```

int func1 (unsigned n)
{
    unsigned i, result = 0;
    for (i = 0; i <= n; i++)
        result += i;
    return result;
}

int func2 (unsigned n)
{
    unsigned i, result = 0;
    for (i = 0; i <= n; i++)
        result += i;
    return result;
}

```

FIG._ 14A

```

module Sum (clock, reset, result_func1, param_func1_n, run_func1, ready_func1, result_func2,
            param_func2_n, run_func2, ready_func2);
    input      clock, reset, run_func1, run_func2;
    input [15:0] param_func1_n, param_func2_n;
    output     ready_func1, ready_func2;
    output [15:0] result_func1, result_func2;
    reg        ready_func1, ready_func2;
    reg      [1:0] state1, state2;
    reg      [15:0] result_func1, result_func2, s_2_func1_i, s_2_func1_result, s_2_func2_i,
                  s_2_func2_result;;
    always @(posedge clock)
    begin
        if (reset)
            state1 = 0;
        else
            begin
                case (state1)
                0: begin
                    if (run_func1)
                        begin
                            ready_func1 = 0;
                            s_2_func1_result = 16'd0;
                            s_2_func1_i = 16'd0;
                            state1 = 1;
                        end
                end
            end
    end

```

FIG._ 14B1

U.S. Patent May 1, 2001 Sheet 41 of 90 US 6,226,776 B1

```

1: state1 = (s_2_func1_i <= param_func1_n) ? 2 : 3;

2: begin
    s_2_func1_result = (s_2_func1_result + s_2_func1_i);
    s_2_func1_i = (s_2_func1_i + 16'd1);
    state1 = 1;
end

3: begin
    result_func1 = s_2_func1_result;
    ready_func1 = 1;
    state1 = 0;
end
endcase
end
end

always @(posedge clock)
begin
if (reset)
    state2 = 0;
else
begin
    case (state2)
    0: begin
        if (run_func2)
begin
        ready_func2 = 0;
        s_2_func2_result = 16'd0;
        s_2_func2_i = 16'd0;
        state2 = 1;
end
end
end

1: state2 = (s_2_func2_i <= param_func2_n) ? 2 : 3;

2: begin
    s_2_func2_result = (s_2_func2_result + s_2_func2_i);
    s_2_func2_i = (s_2_func2_i + 16'd1);
    state2 = 1;
end

3: begin
    result_func2 = s_2_func2_result;
    ready_func2 = 1;
    state2 = 0;
end
endcase
end
end

endmodule

```

FIG._ 14B2

U.S. Patent May 1, 2001 Sheet 42 of 90 US 6,226,776 B1

```
#include <stdio.h>
#include <stdarg.h>
/* Returns the average of a variable list of integers. */

static int average (int first, ...)
{
    int count = 0, sum = 0, i = first;
    va_list marker;

    va_start (marker, first); /* Initialize variable arguments. */

    while (i != -1)
    {
        sum += i;
        count++;
        i = va_arg (marker, int);
    }

    va_end (marker); /* Reset variable arguments. */

    return sum ? (sum / count) : 0;
}

void main ()
{
    /* Call with 3 integers (-1 is used as terminator). */
    printf ("Average is: %d\n", average (2, 3, 4, -1));

    /* Call with 4 integers. */
    printf ("Average is: %d\n", average (5, 7, 9, 11, -1));

    /* Call with just -1 terminator. */
    printf ("Average is: %d\n", average (-1));
}
```

FIG._15A

U.S. Patent May 1, 2001 Sheet 43 of 90 US 6,226,776 B1

```

`define _average_first 5'h1

module udivmod16 (a, b, div, mod);
    input [15:0] a, b;
    output [15:0] div, mod;
    reg [15:0] div, mod, pa0, pa1, pa2, pa3, pa4, pa5, pa6, pa7, pa8, pa9, pa10,
                pa11, pa12, pa13, pa14, pa15, pa16, shb0;
    reg [16:0] shb1;
    reg [17:0] shb2;
    reg [18:0] shb3;
    reg [19:0] shb4;
    reg [20:0] shb5;
    reg [21:0] shb6;
    reg [22:0] shb7;
    reg [23:0] shb8;
    reg [24:0] shb9;
    reg [25:0] shb10;
    reg [26:0] shb11;
    reg [27:0] shb12;
    reg [28:0] shb13;
    reg [29:0] shb14;
    reg [30:0] shb15;

    always @ (a or b)
    begin
        pa0 = a;
        shb15 = b << 15;
        div [15] = pa0 >= shb15;
        pa1 = div [15] ? pa0 - shb15 : pa0;
        shb14 = b << 14;
        div [14] = pa1 >= shb14;
        pa2 = div [14] ? pa1 - shb14 : pa1;
        shb13 = b << 13;
        div [13] = pa2 >= shb13;
        pa3 = div [13] ? pa2 - shb13 : pa2;
        shb12 = b << 12;
        div [12] = pa3 >= shb12;
        pa4 = div [12] ? pa3 - shb12 : pa3;
        shb11 = b << 11;
        div [11] = pa4 >= shb11;
        pa5 = div [11] ? pa4 - shb11 : pa4;
        shb10 = b << 10;
        div [10] = pa5 >= shb10;
        pa6 = div [10] ? pa5 - shb10 : pa5;
        shb9 = b << 9;
        div [9] = pa6 >= shb9;
        pa7 = div [9] ? pa6 - shb9 : pa6;
    end
endmodule

```

FIG._ 15B1

U.S. Patent

May 1, 2001

Sheet 44 of 90

US 6,226,776 B1

```

shb8    = b << 8;
div [ 8] = pa7 >= shb8;
pa8     = div [8] ? pa7 - shb8 : pa7;

shb7    = b << 7;
div [ 7] = pa8 >= shb7;
pa9     = div [7] ? pa8 - shb7 : pa8;

shb6    = b << 6;
div [ 6] = pa9 >= shb6;
pa10   = div [6] ? pa9 - shb6 : pa9;

shb5    = b << 5;
div [ 5] = pa10 >= shb5;
pa11   = div [5] ? pa10 - shb5 : pa10;

shb4    = b << 4;
div [ 4] = pa11 >= shb4;
pa12   = div [4] ? pa11 - shb4 : pa11;

shb3    = b << 3;
div [ 3] = pa12 >= shb3;
pa13   = div [3] ? pa12 - shb3 : pa12;

shb2    = b << 2;
div [ 2] = pa13 >= shb2;
pa14   = div [2] ? pa13 - shb2 : pa13;

shb1    = b << 1;
div [ 1] = pa14 >= shb1;
pa15   = div [1] ? pa14 - shb1 : pa14;

shb0    = b << 0;
div [ 0] = pa15 >= shb0;
pa16   = div [0] ? pa15 - shb0 : pa15;

mod = pa16;
end
endmodule

module sdivmod16 (a, b, div, mod);
  input [15:0] a, b;
  output [15:0] div, mod;
  wire [15:0] ua = a [15] ? - a : a;
  wire [15:0] ub = b [15] ? - b : b;
  wire [15:0] udiv, umod;
  udivmod16 udivmod (ua, ub, udiv, umod);
  wire [15:0] div = (a [15] != b [15]) ? - udiv : udiv;
  wire [15:0] mod = a [15] ? - umod : umod;
endmodule

```

FIG._15B2

U.S. Patent May 1, 2001 Sheet 45 of 90 US 6,226,776 B1

```

module RAM32X1 (O, D, WE, A4, A3, A2, A1, A0)
/* synthesis black_box xc_alias="RAM" */;
output O;
input D, WE, A4, A3, A2, A1, A0;
reg [31:0] d;
wire [4:0] A = { A4, A3, A2, A1, A0 };
assign O = d [A];
always @(A or D or WE)
if (WE)
d [A] = D;
endmodule

module RAM32x1 (a, d, o, we);
input d, we;
input [4:0] a;
output o;
RAM32X1 RAM32X1 (o, d, we, a [4], a [3], a [2], a [1], a [0]);
endmodule

module RAM32x16 (a, d, o, we);
input we;
input [4:0] a;
input [15:0] d;
output [15:0] o;
RAM32x1 U0 (a, d [0], o [0], we);
RAM32x1 U1 (a, d [1], o [1], we);
RAM32x1 U2 (a, d [2], o [2], we);
RAM32x1 U3 (a, d [3], o [3], we);
RAM32x1 U4 (a, d [4], o [4], we);
RAM32x1 U5 (a, d [5], o [5], we);
RAM32x1 U6 (a, d [6], o [6], we);
RAM32x1 U7 (a, d [7], o [7], we);
RAM32x1 U8 (a, d [8], o [8], we);
RAM32x1 U9 (a, d [9], o [9], we);
RAM32x1 U10 (a, d [10], o [10], we);
RAM32x1 U11 (a, d [11], o [11], we);
RAM32x1 U12 (a, d [12], o [12], we);
RAM32x1 U13 (a, d [13], o [13], we);
RAM32x1 U14 (a, d [14], o [14], we);
RAM32x1 U15 (a, d [15], o [15], we);
endmodule

module RAM18x16 (a, d, o, we);
input we;
input [4:0] a;
input [15:0] d;
output [15:0] o;
RAM32x16 U (a, d, o, we);
endmodule

```

FIG._ 15B3

U.S. Patent May 1, 2001 Sheet 46 of 90 US 6,226,776 B1

```

module StdArg (clock, reset, printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48,
               run_main, ready_main );
  input      clock, reset, run_main;
  output     ready_main;
  output [15:0] printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48
  reg        ready_main, we;
  reg [4:0]   state, a, return_state_average;
  reg [15:0]  printf_1_1_line_42, printf_2_1_line_45, printf_3_1_line_48, divmod_a, divmod_b,
               d, result_average, _108_average_count, _108_average_sum, _108_average_i,
               _108_average_marker, _1_16, _2_16, _3_16, _4_16, _5_16, _6_16, _7_16, _8_16,
               _9_16, _10_16, _11_16, _12_16, _13_16, _14_16;
  wire [15:0] div_result, o;
  sdivmod16 sdivmod (divmod_a, divmod_b, div_result, );

  RAM18x16 m (a, d, o, we);
  parameter _average = 1;

  always @(posedge clock)
  begin
    if (reset)
      begin
        we = 0;
        state = 0;
      end
    else
      begin
        case (state)
          0: begin
            if (run_main)
              begin
                ready_main = 0;
                _5_16 = 16'd2;
                a = `_average_first;
                d = _5_16;
                we = 1;
                state = 11;
              end
            end
          _average: begin
            _108_average_count = 16'd0;
            _108_average_sum = 16'd0;
            a = `_average_first;
            state = 5;
          end
          5: begin
            _1_16 = 0;
            _108_average_i = _1_16;
            _108_average_marker = (_average_first + 16'd1);
            state = 2;
          end
          2: state = (_108_average_i != -16'd1) ? 3 : 4;
        endcase
      end
    end
  end
endmodule

```

FIG._ 15B4

U.S. Patent May 1, 2001 Sheet 47 of 90 US 6,226,776 B1

```

3: begin
    _108_average_sum = (_108_average_sum + _108_average_i);
    _108_average_count = (_108_average_count + 16'd1);
    _108_average_marker = (_108_average_marker + 16'd1);
    a = (_108_average_marker - 16'd1);
    state = 6;
end

6: begin
    _2_16 = 0;
    _108_average_i = _2_16;
    state = 2;
end

4: begin
    _108_average_marker = 16'd0;
    state = _108_average_sum ? 7 : 8;
end

7: begin
    divmod_a = _108_average_sum;
    divmod_b = _108_average_count;
    state = 10;
end

10: begin
    _4_16 = div_result;
    _3_16 = _4_16;
    state = 9;
end

8: begin
    _3_16 = 16'd0;
    state = 9;
end

9: begin
    result_average = _3_16;
    state = return_state_average;
end

11: begin
    we = 0;
    _6_16 = 16'd3;
    a = (^_average_first + 16'd1);
    d = _6_16;
    we = 1;
    state = 12;
end

```

FIG._15B5

U.S. Patent May 1, 2001 Sheet 48 of 90 US 6,226,776 B1

```

12: begin
    we = 0;
    _7_16 = 16'd4;
    a = (_average_first + 16'd2);
    d = _7_16;
    we = 1;
    state = 13;
end

13: begin
    we = 0;
    _8_16 = -16'd1;
    a = (_average_first + 16'd3);
    d = _8_16;
    we = 1;
    state = 14;
end

14: begin
    we = 0;
    return_state_average = 15;
    state = _average;
end

15: begin
    printf_1_1_line_42 = result_average;
    // User Verilog code
    $write ("Average is: %d\n", printf_1_1_line_42);
    // End of user Verilog code
    _9_16 = 16'd5;
    a = `_average_first;
    d = _9_16;
    we = 1;
    state = 16;
end

16: begin
    we = 0;
    _10_16 = 16'd7;
    a = (_average_first + 16'd1);
    d = _10_16;
    we = 1;
    state = 17;
end

17: begin
    we = 0;
    _11_16 = 16'd9;
    a = (_average_first + 16'd2);
    d = _11_16;
    we = 1;
    state = 18;
end

```

FIG._ 15B6

U.S. Patent

May 1, 2001

Sheet 49 of 90

US 6,226,776 B1

```

18: begin
    we = 0;
    _12_16 = 16'd11;
    a = (^_average_first + 16'd3);
    d = _12_16;
    we = 1;
    state = 19;
end

19: begin
    we = 0;
    _13_16 = -16'd1;
    a = (^_average_first + 16'd4);
    d = _13_16;
    we = 1;
    state = 20;
end

20: begin
    we = 0;
    return_state_average = 21;
    state = _average;
end

21: begin
    printf_2_1_line_45 = result_average;
    // User Verilog code
    $write ("Average is: %d\n", printf_2_1_line_45);
    // End of user Verilog code
    _14_16 = -16'd1;
    a = `^_average_first;
    d = _14_16;
    we = 1;
    state = 22;
end

22: begin
    we = 0;
    return_state_average = 23;
    state = _average;
end

23: begin
    printf_3_1_line_48 = result_average;
    // User Verilog code
    $write ("Average is: %d\n", printf_3_1_line_48);
    // End of user Verilog code
    ready_main = 1;
    state = 0;
end

default: ;
endcase
end
end
endmodule

```

FIG._15B7

U.S. Patent May 1, 2001 Sheet 50 of 90 US 6,226,776 B1

```
#include <stdio.h>

static void MoveRing (int nRings, int Tower1, int Tower2, int Tower3)
{
    if (nRings == 0)
        return;

    MoveRing (nRings - 1, Tower1, Tower3, Tower2);
    printf ("%d -> %d\n", Tower1, Tower2);
    MoveRing (nRings - 1, Tower3, Tower2, Tower1);
}

void Hanoi (int nRings)
{
    MoveRing (nRings, 1, 2, 3);
}

#ifndef __SYNETRY__
void main (void)
{
    printf ("Five rings Hanoi Towers\n");
    Hanoi (5);
}
#endif
```

FIG._ 16A

U.S. Patent

May 1, 2001

Sheet 51 of 90

US 6,226,776 B1

```

`define recursion_stack 8'h1
module Hanoi (clock, reset, a, d, o, we, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2,
              param_Hanoi_nRings, run_Hanoi, ready_Hanoi );
  input      clock, reset, run_Hanoi;
  input [ 7:0] o, param_Hanoi_nRings;
  output     we, ready_Hanoi;
  output [ 7:0] a, d, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2;
  reg        we, ready_Hanoi;
  reg [ 4:0] state, return_state_MoveRing;
  reg [ 7:0] a, d, printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2, recursion_stack_pointer,
             _MoveRing_nRings, _MoveRing_Tower1, _MoveRing_Tower2, _MoveRing_Tower3,
             _1_8, _2_8, _3_8, _4_8, _5_8, _6_8, _7_8, _8_8, _9_8, _10_8, _11_8, _12_8;
  parameter _MoveRing = 1;

  always @(posedge clock)
  begin
    if (reset)
      begin
        we = 0;
        recursion_stack_pointer = `recursion_stack;
        state = 0;
      end
    else
      begin
        case (state)
          0: begin
            if (run_Hanoi)
              begin
                ready_Hanoi = 0;
                _MoveRing_nRings = param_Hanoi_nRings;
                _MoveRing_Tower1 = 8'd1;
                _MoveRing_Tower2 = 8'd2;
                _MoveRing_Tower3 = 8'd3;
                return_state_MoveRing = 26;
                state = _MoveRing;
              end
            end
          _MoveRing: state = (_MoveRing_nRings == 8'd0) ? 3 : 2;
          2: begin
            _1_8 = (_MoveRing_nRings - 8'd1);
            _2_8 = _MoveRing_Tower1;
            _3_8 = _MoveRing_Tower3;
            _4_8 = _MoveRing_Tower2;
            a = recursion_stack_pointer;
            d = _MoveRing_nRings;
            we = 1;
            state = 4;
          end
        end
      end
  end

```

FIG._ 16B1

U.S. Patent

May 1, 2001

Sheet 52 of 90

US 6,226,776 B1

```

4: begin
    we = 0;
    a = recursion_stack_pointer + 1;
    d = _MoveRing_Tower1;
    we = 1;
    state = 5;
end

5: begin
    we = 0;
    a = recursion_stack_pointer + 2;
    d = _MoveRing_Tower2;
    we = 1;
    state = 6;
end

6: begin
    we = 0;
    a = recursion_stack_pointer + 3;
    d = _MoveRing_Tower3;
    we = 1;
    state = 7;
end

7: begin
    we = 0;
    _9_8 = return_state_MoveRing;
    a = recursion_stack_pointer + 4;
    d = _9_8;
    we = 1;
    state = 8;
end

8: begin
    we = 0;
    recursion_stack_pointer = recursion_stack_pointer + 5;
    _MoveRing_nRings = _1_8;
    _MoveRing_Tower1 = _2_8;
    _MoveRing_Tower2 = _3_8;
    _MoveRing_Tower3 = _4_8;
    return_state_MoveRing = 9;
    state = _MoveRing;
end

9: begin
    recursion_stack_pointer = recursion_stack_pointer - 5;
    a = recursion_stack_pointer;
    state = 10;
end

10: begin
    _MoveRing_nRings = 0;
    a = recursion_stack_pointer + 1;
    state = 11;
end

```

FIG._ 16B2

U.S. Patent May 1, 2001 Sheet 53 of 90 US 6,226,776 B1

```

11: begin
    _MoveRing_Tower1 = 0;
    a = recursion_stack_pointer + 2;
    state = 12;
end

12: begin
    _MoveRing_Tower2 = 0;
    a = recursion_stack_pointer + 3;
    state = 13;
end

13: begin
    _MoveRing_Tower3 = 0;
    a = recursion_stack_pointer + 4;
    state = 14;
end

14: begin
    _10_8 = 0;
    return_state_MoveRing = _10_8;
    printf_1_1_line_22_Tower1 = _MoveRing_Tower1;
    printf_1_2_line_22_Tower2 = _MoveRing_Tower2;
    // User Verilog code
    $write ("%d -> %d\n", printf_1_1_line_22_Tower1, printf_1_2_line_22_Tower2);
    // End of user Verilog code
    _5_8 = (_MoveRing_nRings - 8'd1);
    _6_8 = _MoveRing_Tower3;
    _7_8 = _MoveRing_Tower2;
    _8_8 = _MoveRing_Tower1;
    a = recursion_stack_pointer;
    d = _MoveRing_nRings;
    we = 1;
    state = 15;
end

15: begin
    we = 0;
    a = recursion_stack_pointer + 1;
    d = _MoveRing_Tower1;
    we = 1;
    state = 16;
end

16: begin
    we = 0;
    a = recursion_stack_pointer + 2;
    d = _MoveRing_Tower2;
    we = 1;
    state = 17;
end

```

FIG._ 16B3

U.S. Patent

May 1, 2001

Sheet 54 of 90

US 6,226,776 B1

```

17: begin
    we = 0;
    a = recursion_stack_pointer + 3;
    d = _MoveRing_Tower3;
    we = 1;
    state = 18;
end

18: begin
    we = 0;
    _11_8 = return_state_MoveRing;
    a = recursion_stack_pointer + 4;
    d = _11_8;
    we = 1;
    state = 19;
end

19: begin
    we = 0;
    recursion_stack_pointer = recursion_stack_pointer + 5;
    _MoveRing_nRings = _5_8;
    _MoveRing_Tower1 = _6_8;
    _MoveRing_Tower2 = _7_8;
    _MoveRing_Tower3 = _8_8;
    return_state_MoveRing = 20;
    state = _MoveRing;
end

20: begin
    recursion_stack_pointer = recursion_stack_pointer - 5;
    a = recursion_stack_pointer;
    state = 21;
end

21: begin
    _MoveRing_nRings = 0;
    a = recursion_stack_pointer + 1;
    state = 22;
end

22: begin
    _MoveRing_Tower1 = 0;
    a = recursion_stack_pointer + 2;
    state = 23;
end

23: begin
    _MoveRing_Tower2 = 0;
    a = recursion_stack_pointer + 3;
    state = 24;
end

```

FIG._ 16B4

U.S. Patent May 1, 2001 Sheet 55 of 90 US 6,226,776 B1

```
24: begin
    _MoveRing_Tower3 = 0;
    a = recursion_stack_pointer + 4;
    state = 25;
end

25: begin
    _12_8 = 0;
    return_state_MoveRing = _12_8;
    state = 3;
end

3: state = return_state_MoveRing;

26: begin
    ready_Hanoi = 1;
    state = 0;
end

default: ;
endcase
end
end
```

endmodule

FIG._16B5

U.S. Patent May 1, 2001 Sheet 56 of 90 US 6,226,776 B1

```
#include <stdio.h>

void f1 () {
    printf ("%d\n", 1);
}

void f2 () {
    printf ("%d\n", 2);
}

void main (n)
{
    void (* f) ();
    if (n == 1 ? f1 : f2);
    f ();
}
```

FIG._ 17A

```
module PtrFtn (clock, reset, printf_1_1_line_19, printf_2_1_line_24, param_main_n, run_f1, ready_f1,
              run_f2, ready_f2, run_main, ready_main);
    input      clock, reset, run_f1, run_f2, run_main;
    input [15:0] param_main_n;
    output     ready_f1, ready_f2, ready_main;
    output [15:0] printf_1_1_line_19, printf_2_1_line_24;
    reg        ready_f1, ready_f2, ready_main;
    reg [1:0]   state, return_state_f1, return_state_f2;
    reg [15:0]  printf_1_1_line_19, printf_2_1_line_24, _110_main_f;
    parameter _f1 = 1, _f2 = 2;

    always @(posedge clock)
    begin
        if (reset)
        begin
            printf_1_1_line_19 = 0;
            printf_2_1_line_24 = 0;
            _110_main_f      = 0;
            ready_f1          = 0;
            return_state_f1   = 0;
            ready_f2          = 0;
            return_state_f2   = 0;
            ready_main         = 0;
            state = 0;
        end
    end
```

FIG._ 17B1

U.S. Patent May 1, 2001 Sheet 57 of 90 US 6,226,776 B1

```

else
begin
  case (state)
    0: begin
      if (run_f1)
        begin
          ready_f1 = 0;
          return_state_f1 = 0;
          state = _f1;
        end
      else if (run_f2)
        begin
          ready_f2 = 0;
          return_state_f2 = 0;
          state = _f2;
        end
      else if (run_main)
        begin
          ready_main = 0;
          _110_main_f = ((param_main_n == 16'd1) ? _f1 : _f2);
          state = _110_main_f;
        end
      end
    _f1: begin
      printf_1_1_line_19 = 16'd1;
      // User Verilog code
      $write ("%d\n", printf_1_1_line_19);
      // End of user Verilog code
      ready_f1 = 1;
      state = return_state_f1;
    end
    _f2: begin
      printf_2_1_line_24 = 16'd2;
      // User Verilog code
      $write ("%d\n", printf_2_1_line_24);
      // End of user Verilog code
      ready_f2 = 1;
      state = return_state_f2;
    end
    3: begin
      ready_main = 1;
      state = 0;
    end
    default: ;
  endcase
end
end
endmodule

```

FIG._17B2

U.S. Patent May 1, 2001 Sheet 58 of 90 US 6,226,776 B1

```
int sum (int array [], int size)
{
    int i, sum = 0;

    for (i = 0; i < size; i++)
        sum += array [i];

    return sum;
}

int main ()
{
    int i;
    int array [10];
    int size = sizeof (array) / sizeof (*array);

    array [0] = 1;
    array [1] = 2;
    array [2] = 3;

    for (i = 3; i < size; i++)
        array [i] = i * 2;

    return sum (array, size);
}
```

FIG._18A

U.S. Patent May 1, 2001 Sheet 59 of 90 US 6,226,776 B1

```

// Global macro definitions
`define UNIT_MAIN      16'h5000
`define UNIT_SIZE       16
`define ADDRESS_SIZE   16
`define DATA_SIZE       32

// Local macro definitions
`define STATE_SIZE      5
`define WORD_SIZE        32
`define WORD_MEMORY_SIZE 15
`define MEMORY_SIZE     480

// External functions defined in the translated C file
`define _sum            32'h50000001
`define offset_sum       1
`define var_sum          memory [1]
`define state_sum        5'd1
`define _sum_array       32'h50000002
`define offset_sum_array 2
`define var_sum_array    memory [2]
`define _sum_size         32'h50000003
`define offset_sum_size  3
`define var_sum_size     memory [3]
`define _main             32'h50000004
`define offset_main       4
`define var_main          memory [4]
`define state_main        5'd2

// Static, auto and register variables located in module's internal memory
`define main_3_main_array 32'h50000005
`define offset_main_3_main_array 5

module main (reset, clock, master, in_unit, in_address, in_data, in_read, in_call, out_unit,
            out_address, out_data, out_read, out_call);
    input      reset, clock, in_call, in_read;
    output     master, out_read, out_call;
    reg        master, out_read, out_call;

    input [`UNIT_SIZE - 1 : 0] in_unit;
    input [`ADDRESS_SIZE - 1 : 0] in_address;
    input [`DATA_SIZE - 1 : 0] in_data;

    output [`UNIT_SIZE - 1 : 0] out_unit;
    output [`ADDRESS_SIZE - 1 : 0] out_address;
    output [`DATA_SIZE - 1 : 0] out_data;

    reg   [`UNIT_SIZE - 1 : 0] out_unit;
    reg   [`ADDRESS_SIZE - 1 : 0] out_address;
    reg   [`DATA_SIZE - 1 : 0] out_data;

// Module data
    reg   [`STATE_SIZE - 1 : 0] state;
    reg   [`STATE_SIZE - 1 : 0] return_state;
    reg   [`WORD_SIZE - 1 : 0] memory [0 : `WORD_MEMORY_SIZE - 1];

```

FIG._ 18B1

U.S. Patent

May 1, 2001

Sheet 60 of 90

US 6,226,776 B1

```

// Predefined states parameter [4:0]
    init      = 5'd0,
    idle      = 5'd20,
    read      = 5'd21,
    write     = 5'd22,
    intercall = 5'd23,
    intercall2 = 5'd24,
    intercall3 = 5'd25,
    answer    = 5'd26,
    answer2   = 5'd27;
// Function result registers
    reg [31:0] result__sum, result__main;
// Static, auto and register variables located on registers
    reg [31:0] main_2_sum_i, main_2_sum_sum, main_3_main_i, main_3_main_size;
// Temporary registers
    reg [31 : 0] _6_32, _8_32, _9_32, _11_32;      reg [ 0 : 0] _7_1, _10_1;
always @ (posedge reset or posedge clock) // Clock cycle
begin
    if (reset)
        begin
            master = 0;
            state = idle;
        end
    else
        begin
            case (state)
                idle: begin
                    if (in_unit == `UNIT_MAIN)
                        begin
                            if (in_call)
                                begin
                                    out_unit = in_unit;
                                    out_address = in_address;
                                    out_data = answer;
                                    return_state = in_address;
                                    state = write;
                                end
                            else
                                begin
                                    out_unit = in_unit;
                                    out_address = in_address;
                                    return_state = answer;
                                    if (in_read)
                                        state = read;
                                    else
                                        begin
                                            out_data = in_data;
                                            state = write;
                                        end
                                end
                            end
                        end
                    end
                end
            end
        end
end

```

FIG._ 18B2

U.S. Patent May 1, 2001 Sheet 61 of 90 US 6,226,776 B1

```

init: begin
    master = 1;
    state = answer;
end

`state__sum: begin
    main_2_sum_sum = 32'd0;
    main_2_sum_i = 32'd0;
    state = `STATE_SIZE'd11;
end

`STATE_SIZE'd11: begin
    _6_32 = (main_2_sum_i - `var__sum__size);
    _7_1 = _6_32 [31] == 1'b1;
    state = _7_1 ? `STATE_SIZE'd12 : `STATE_SIZE'd13;
end

`STATE_SIZE'd12: begin
    { out_unit, out_address } = (`var__sum__array + main_2_sum_i);
    return_state = `STATE_SIZE'd17;
    state = read;
end

`STATE_SIZE'd17: begin
    _8_32 = out_data;
    main_2_sum_sum = (main_2_sum_sum + _8_32);
    main_2_sum_i = (main_2_sum_i + 32'd1);
    state = `STATE_SIZE'd11;
end

`STATE_SIZE'd13: begin
    result__sum = main_2_sum_sum;
    { out_unit, out_address } = `_sum;
    out_data = result__sum;
    master = 1;
    state = `var__sum;
end

`state_main: begin
    main_3_main_size = 32'd10;
    memory [main_3_main_array - (UNIT_MAIN << `ADDRESS_SIZE)] = 32'd1;
    memory [(main_3_main_array + 32'd1) - (UNIT_MAIN << `ADDRESS_SIZE)] = 32'd2;
    memory [(main_3_main_array + 32'd2) - (UNIT_MAIN << `ADDRESS_SIZE)] = 32'd3;
    main_3_main_i = 32'd3;
    state = `STATE_SIZE'd14;
end

`STATE_SIZE'd14: begin
    _9_32 = (main_3_main_i - main_3_main_size);
    _10_1 = _9_32 [31] == 1'b1;
    state = _10_1 ? `STATE_SIZE'd15 : `STATE_SIZE'd16;
end

```

FIG._ 18B3

U.S. Patent May 1, 2001 Sheet 62 of 90 US 6,226,776 B1

```

`STATE_SIZE'd15: begin
    _11_32 = (main_3_main_i << 1'd1);
    { out_unit, out_address } = (^main_3_main_array + main_3_main_i);
    out_data = _11_32;
    return_state = `STATE_SIZE'd18;
    state = write;
end

`STATE_SIZE'd18: begin
    main_3_main_i = (main_3_main_i + 32'd1);
    state = `STATE_SIZE'd14;
end

`STATE_SIZE'd16: begin
    `var_sum_array = `main_3_main_array;
    `var_sum_size = main_3_main_size;
    `var_sum = `STATE_SIZE'd19;
    state = `state_sum;
end

`STATE_SIZE'd19: begin
    result_main = result_sum;
    { out_unit, out_address } = `_main;
    out_data = result_main;
    master = 1;
    state = `var_main;
end

read: begin
    if (out_unit != `UNIT_MAIN)
        begin
            out_read = 1;
            out_call = 0;
            master = 1;
            state' = intercall;
        end
    else
        begin
            out_data = memory [out_address];
            if (return_state == answer)
                master = 1;
            state = return_state;
        end
end.

```

FIG._ 18B4

U.S. Patent

May 1, 2001

Sheet 63 of 90

US 6,226,776 B1

```

write: begin
    if (out_unit != `UNIT_MAIN)
        begin
            out_read = 0;
            out_call = 0;
            master   = 1;
            state    = intercall;
        end
    else
        begin
            memory [out_address] = out_data;
            if (return_state == answer)
                master = 1;
            state = return_state;
        end
    end
    intercall: begin
        state = intercall2;
    end
    intercall2: begin
        master = 0;
        state = intercall3;
    end
    intercall3: begin
        if ((in_unit, in_address) == (out_unit, out_address))
            begin
                out_data = in_data;
                state = return_state;
            end
    end
    answer: begin
        state = answer2;
    end
    answer2: begin
        master = 0;
        state = idle;
    end
    default:
        state = idle;
    endcase
end
end
endmodule

```

FIG._ 18B5

U.S. Patent

May 1, 2001

Sheet 64 of 90

US 6,226,776 B1

```

int data_in;
int out1;
int out2;
int data_out;

void pipeline_stage_1 ()
{
    while (data_in != 0)
        out1 = data_in + 1;

    if (data_in == 1)
        while (data_in == 1)
            out1 = 2;
}

void pipeline_stage_2 ()
{
    while (out1 == 0)
        out2 = 0;

    while (out1 != 0)
        out2 = out1 + 1;
}

void pipeline_stage_3 ()
{
    if (out2 == 1)
        while (out2 == 1)
            data_out = 2;

    while (out2 != 0)
        data_out = out2 + 1;
}

#ifndef __SYNETRY__
#include <stdio.h>

void main ()
{
    for (;;)
    {
        scanf ("%d", & data_in);
        pipeline_stage_1 ();
        pipeline_stage_2 ();
        pipeline_stage_3 ();
        printf ("%d", data_out);
    }
}
#endif

```

FIG._19A

U.S. Patent May 1, 2001 Sheet 65 of 90 US 6,226,776 B1

```

module Pipeline(clock,reset,_data_in,_data_out,run_pipeline_stage_1,ready_pipeline_stage_1,
               run_pipeline_stage_2,ready_pipeline_stage_2,run_pipeline_stage_3,ready_pipeline_stage_3);
  input   clock, reset, run_pipeline_stage_1, run_pipeline_stage_2, run_pipeline_stage_3;
  input [15:0] _data_in;
  output  ready_pipeline_stage_1, ready_pipeline_stage_2, ready_pipeline_stage_3;
  output [15:0] _data_out;
  reg      ready_pipeline_stage_1, ready_pipeline_stage_2, ready_pipeline_stage_3;
  reg     [2:0] state1, state2, state3;
  reg     [15:0] _data_out, _out1, _out2;

  always @(posedge clock)
  begin
    if (reset)
      state1 = 0;
    else
      begin
        case (state1)

          0: begin
            if (run_pipeline_stage_1)
              begin
                ready_pipeline_stage_1 = 0;
                state1 = 1;
              end
            end

          1: state1 = (_data_in != 16'd0) ? 2 : 3;

          2: begin
            _out1 = (_data_in + 16'd1);
            state1 = 1;
          end

          3: state1 = (_data_in == 16'd1) ? 4 : 6;

          4: state1 = (_data_in == 16'd1) ? 5 : 6;

          5: begin
            _out1 = 16'd2;
            state1 = 4;
          end

          6: begin
            ready_pipeline_stage_1 = 1;
            state1 = 0;
          end

          default: ;

        endcase
      end
    end
  
```

FIG._ 19B1

U.S. Patent May 1, 2001 Sheet 66 of 90 US 6,226,776 B1

```

always @(posedge clock)
begin
    if (reset)
        state2 = 0;
    else
        begin
            case (state2)

                0: begin
                    if (run_pipeline_stage_2)
                        begin
                            ready_pipeline_stage_2 = 0;
                            state2 = 1;
                        end
                    end

                1: state2 = (_out1 == 16'd0) ? 2 : 3;

                2: begin
                    _out2 = 16'd0;
                    state2 = 1;
                end

                3: state2 = (_out1 != 16'd0) ? 4 : 5;

                4: begin
                    _out2 = (_out1 + 16'd1);
                    state2 = 3;
                end

                5: begin
                    ready_pipeline_stage_2 = 1;
                    state2 = 0;
                end

                default: ;

            endcase
        end
    end

```

FIG._ 19B2

U.S. Patent May 1, 2001 Sheet 67 of 90 US 6,226,776 B1

```

always @(posedge clock)
begin
    if (reset)
        state3 = 0;
    else
        begin
            case (state3)

                0: begin
                    if (run_pipeline_stage_3)
                        begin
                            ready_pipeline_stage_3 = 0;
                            state3 = (_out2 == 16'd1) ? 1 : 3;
                        end
                    end

                1: state3 = (_out2 == 16'd1) ? 2 : 3;

                2: begin
                    _data_out = 16'd2;
                    state3 = 1;
                end

                3: state3 = (_out2 != 16'd0) ? 4 : 5;

                4: begin
                    _data_out = (_out2 + 16'd1);
                    state3 = 3;
                end

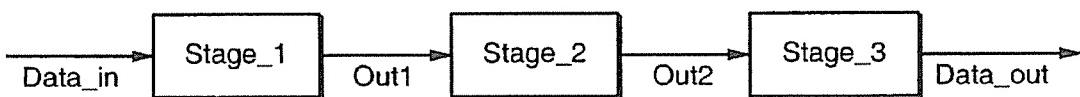
                5: begin
                    ready_pipeline_stage_3 = 1;
                    state3 = 0;
                end

                default: ;

            endcase
        end
    end

endmodule

```

FIG._ 19B3***FIG._ 19C***

U.S. Patent May 1, 2001 Sheet 68 of 90 US 6,226,776 B1

```
#include <stdio.h>

typedef enum { true = 1, false = 0 } bool;

void main ()
{
    bool bLightIsOn = true;

    printf ("Light.c - C program working on APS-X84 FPGA test card\n");

    for (;;)
    {
        int counter;

        printf
        (
            "Enter the number of clock ticks"
            " to wait before the light change: "
        );

        scanf ("%d", & counter);

        if (counter < 0)
            counter = 0;

        while (counter -- != 0)
            printf ("Waiting...\n");

        bLightIsOn = ! bLightIsOn;

        printf ("Light is: %d\n", bLightIsOn);
    }
}
```

FIG._20A

U.S. Patent May 1, 2001 Sheet 69 of 90 US 6,226,776 B1

```

module Light (clock, reset, scanf_0_1_line_33_counter, printf_4_1_line_43_bLightIsOn,
              run_main, ready_main );
  input      clock, reset, run_main;
  input [ 7:0] scanf_0_1_line_33_counter;

  output     ready_main;
  output [ 1:0] printf_4_1_line_43_bLightIsOn;
  reg        ready_main, _1_1;
  reg [ 1:0] printf_4_1_line_43_bLightIsOn, _3_main_bLightIsOn, _3_2;
  reg [ 2:0] state;
  reg [ 7:0] _4_main_counter, _2_8;

  always @(posedge clock)
  begin
    if (reset)
      state = 0;
    else
      begin
        case (state)

          0: begin
            if (run_main)
              begin
                ready_main = 0;
                _3_main_bLightIsOn = 2'd1;
                // User Verilog code
                $write ("Light.c - C program working on APS-X84 FPGA test card\n");
                // End of user Verilog code
                state = 1;
              end
            end
          end

          1: begin
            // User Verilog code
            $write ("Enter the number of clock ticks to wait before the light change: ");
            // End of user Verilog code
            _4_main_counter = scanf_0_1_line_33_counter;
            _1_1 = _4_main_counter [7] == 1'b1;
            if (_1_1)
              _4_main_counter = 8'd0;
            state = 2;
          end

          2: begin
            _2_8 = _4_main_counter;
            _4_main_counter = (_2_8 - 8'd1);
            state = (_2_8 != 8'd0) ? 3 : 4;
          end
        endcase
      end
  end
endmodule

```

FIG._20B1

U.S. Patent May 1, 2001 Sheet 70 of 90 US 6,226,776 B1

```

3: begin
    // User Verilog code
    $write ("Waiting...\n");
    // End of user Verilog code
    state = 2;
end

4: begin
    _3_2 = (! _3_main_bLightIsOn);
    _3_main_bLightIsOn = _3_2;
    printf_4_1_line_43_bLightIsOn = _3_main_bLightIsOn;
    // User Verilog code
    $write ("Light is: %d\n", printf_4_1_line_43_bLightIsOn);
    // End of user Verilog code
    state = 1;
end

default: ;

endcase
end
end

endmodule

//  APSX84.v - Verilog wrapper module which instantiates module Light.v.
//      This design is intended to run on APS-X84 board.
//

module APSX84 ( clock, clock_enable, reset, data, LCD);

    input      clock      /* synthesis xc_loc=P24 */;
    input      clock_enable /* synthesis xc_loc=P3 */;
    input      reset      /* synthesis xc_loc=P4 */;
    input [5:0] data       /* synthesis xc_loc="P10,P9,P8,P7,P6,P5" */;
    output     LCD        /* synthesis xc_loc=P35 */;

    wire      internal_clock = clock & clock_enable;
    wire [7:0] data8 = ( data, 2'b0 );
    wire [1:0] LCD2;
    wire      LCD = LCD2 [0];
    wire      run_main = 1;

    Light Light ( internal_clock, reset, data8, LCD2, run_main,);

endmodule

```

FIG._20B2

U.S. Patent May 1, 2001 Sheet 71 of 90 US 6,226,776 B1

```

int sum (int n)
{
    int i, sum = 0;
    for (i = 0; i != n; i++)
        sum += i;
    return sum;
}

```

FIG._21A

```

module Sum (clock, reset, result_sum, param_sum_n, run_sum, ready_sum );
    input      clock, reset, run_sum;
    input [15:0] param_sum_n;
    output     ready_sum;
    output [15:0] result_sum;
    reg       ready_sum;
    reg [1:0]   state;
    reg [15:0] result_sum, s_2_sum_i, s_2_sum_sum;

    always @(posedge clock)
    begin
        if (reset)
            state = 0;
        else
        begin
            case (state)
                0: begin
                    if (run_sum)
                        begin
                            ready_sum = 0;
                            s_2_sum_sum = 16'd0;
                            s_2_sum_i = 16'd0;
                            state = 1;
                        end
                end
                1: state = (s_2_sum_i != param_sum_n) ? 2 : 3;
                2: begin
                    s_2_sum_sum = (s_2_sum_sum + s_2_sum_i);
                    s_2_sum_i = (s_2_sum_i + 16'd1);
                    state = 1;
                end
                3: begin
                    result_sum = s_2_sum_sum;
                    ready_sum = 1;
                    state = 0;
                end
                default: ;
            endcase
        end
    end
endmodule

```

FIG._21B

U.S. Patent

May 1, 2001

Sheet 72 of 90

US 6,226,776 B1

```

library IEEE;
use IEEE.STD_Logic_1164.all;
use IEEE.Numeric_Std.all;

entity synetry_sum is
port( clock      : in std_logic;
      reset      : in boolean;
      run_sum    : in boolean;
      param_sum_n : in integer range 0 to 65535;
      result_sum : out integer range 0 to 65535;
      ready_sum  : out boolean );
end entity synetry_sum;

architecture RTL of synetry_sum is
begin
  process_synetry_sum : process
    variable state      : integer range 0 to 3;
    variable s_2_sum_i  : integer range 0 to 65535;
    variable s_2_sum_sum : integer range 0 to 65535;
begin
  begin
    wait until clock'event and clock = '1';
    if (reset) then
      state := 0;
    else
      case (state)is
        when 0 =>
          if (run_sum)then
            ready_sum <= false;
            s_2_sum_sum := 0;
            s_2_sum_i := 0;
            state := 1;
          end if;
        when 1 =>
          if (s_2_sum_i /= param_sum_n) then
            state := 2;
          else
            state := 3;
          end if;
        when 2 =>
          s_2_sum_sum := s_2_sum_sum + s_2_sum_i;
          s_2_sum_i := s_2_sum_i + 1;
          state := 1;
        when 3 =>
          result_sum <= s_2_sum_sum;
          ready_sum <= true;
          state := 0;
      end case;
    end if;
    wait;
  end process process_synetry_sum;
end architecture RTL;

```

FIG._21C

U.S. Patent

May 1, 2001

Sheet 73 of 90

US 6,226,776 B1

```
int read (int * p)
{
    return * p;
}
```

FIG._22A

```
module Read(clock,reset,a,d,o,we,result_read,param_read_p,run_read,ready_read);
    input      clock, reset, run_read;
    input [ 3:0] o, param_read_p;
    output     we, ready_read;
    output [ 3:0] a, d, result_read;
    reg        we, ready_read, state;
    reg [ 3:0] a, d, result_read;

    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
                0:begin
                    if (run_read)
                    begin
                        ready_read = 0;
                        a = param_read_p;
                        state = 1;
                    end
                end
                1:begin
                    _1_4 = o;
                    result_read = _1_4;
                    ready_read = 1;
                    state = 0;
                end
                default: ;
            endcase
        end
    end
endmodule
```

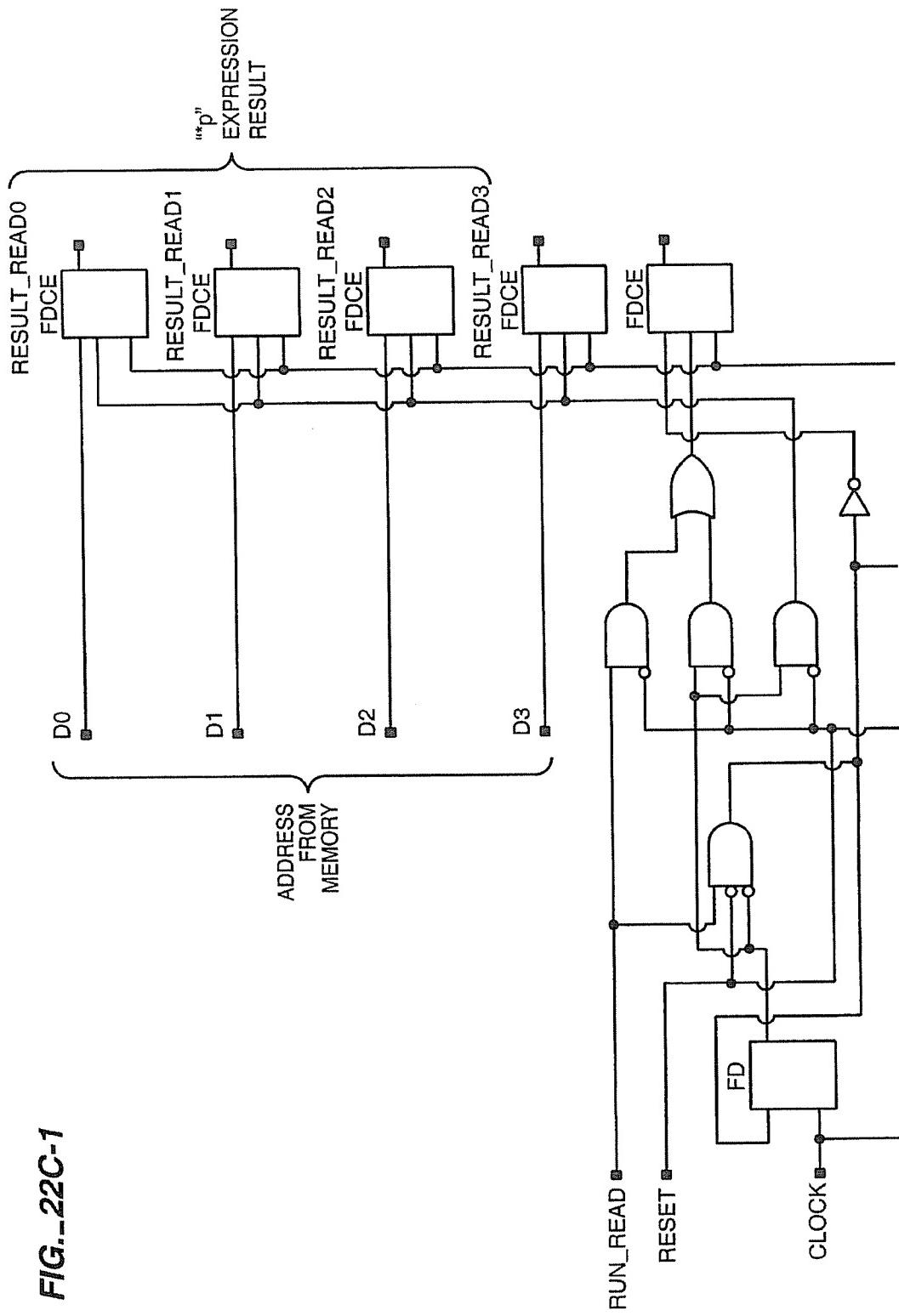
FIG._22B

U.S. Patent

May 1, 2001

Sheet 74 of 90

US 6,226,776 B1

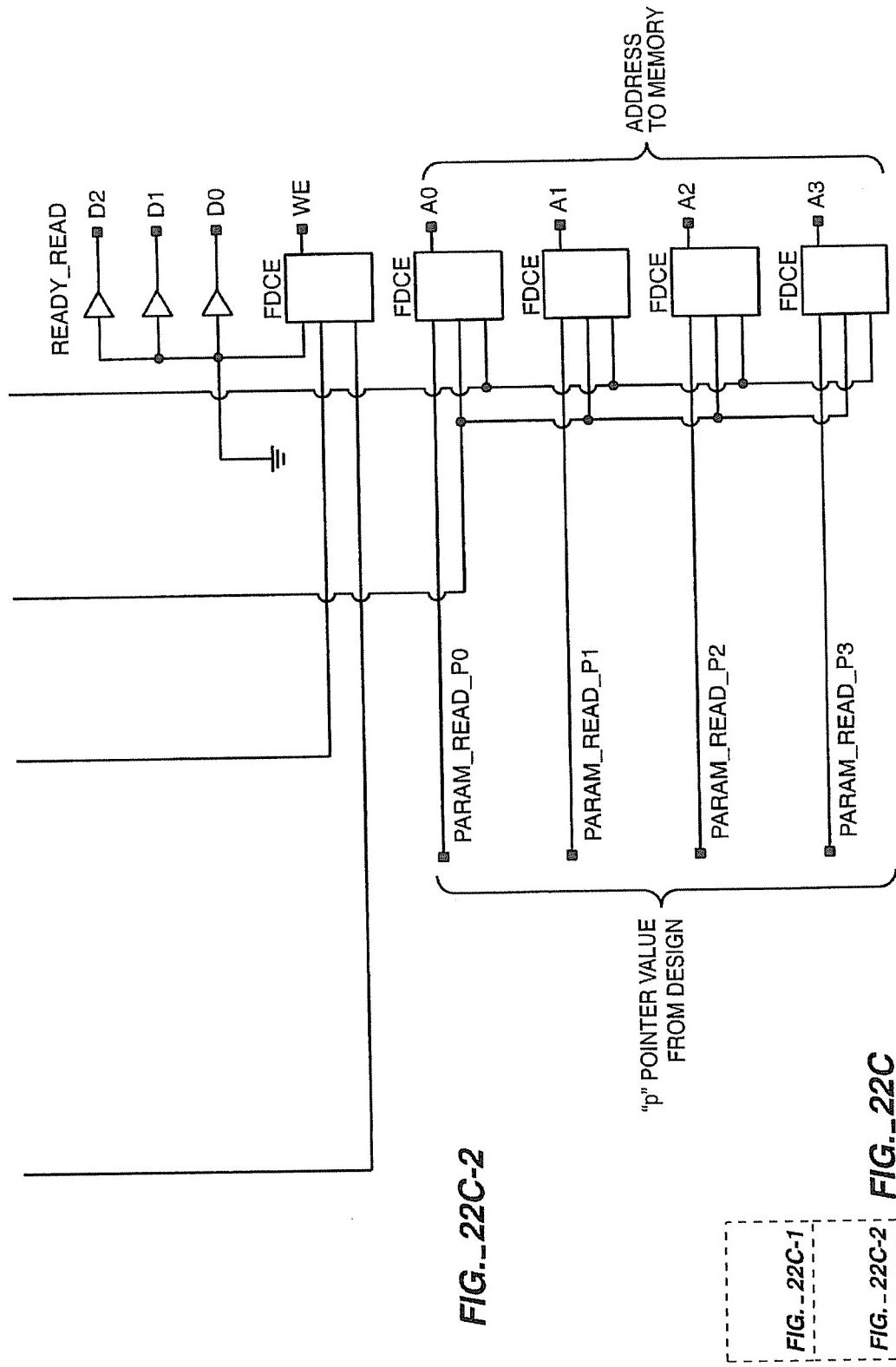


U.S. Patent

May 1, 2001

Sheet 75 of 90

US 6,226,776 B1



U.S. Patent

May 1, 2001

Sheet 76 of 90

US 6,226,776 B1

```
void write (int * p, int d)
{
    * p = d;
}
```

FIG._23A

```
module Write(clock,reset,a,d,o,we,param_write_p,param_write_d,run_write,ready_write);
    input      clock, reset, run_write;
    input [ 3:0] o, param_write_p, param_write_d;
    output     we, ready_write;
    output [ 3:0] a, d;
    reg        we, ready_write, state;
    reg [ 3:0] a, d;

    always @(posedge clock)
    begin
        if (reset)
            begin
                we = 0;
                state = 0;
            end
        else
            begin
                case (state)
                    0:begin
                        if (run_write)
                            begin
                                ready_write = 0;
                                a = param_write_p;
                                d = param_write_d;
                                we = 1;
                                state = 1;
                            end
                    end
                    1:begin
                        we = 0;
                        ready_write = 1;
                        state = 0;
                    end
                    default: ;
                endcase
            end
    end

endmodule
```

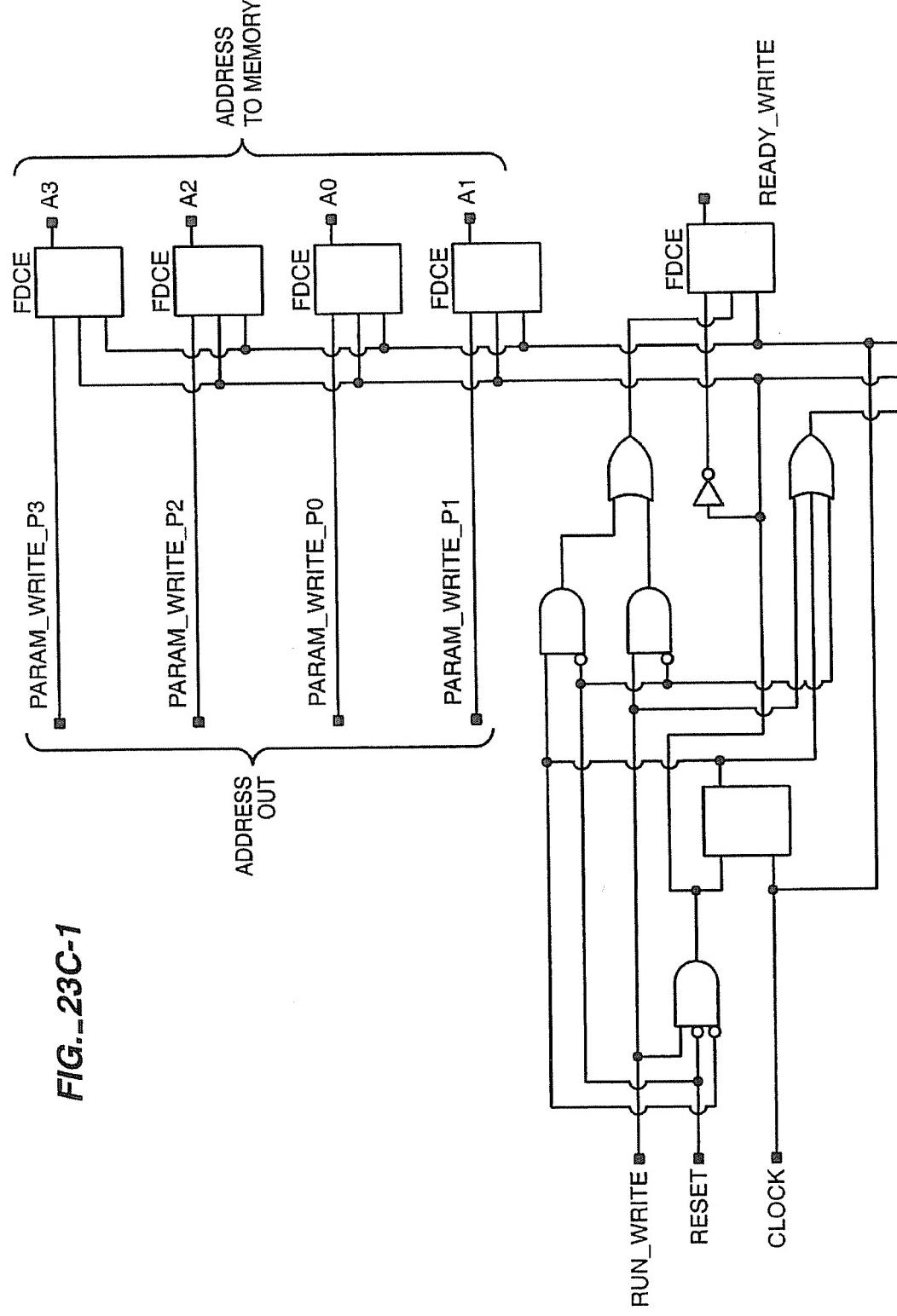
FIG._23B

U.S. Patent

May 1, 2001

Sheet 77 of 90

US 6,226,776 B1



U.S. Patent

May 1, 2001

Sheet 78 of 90

US 6,226,776 B1

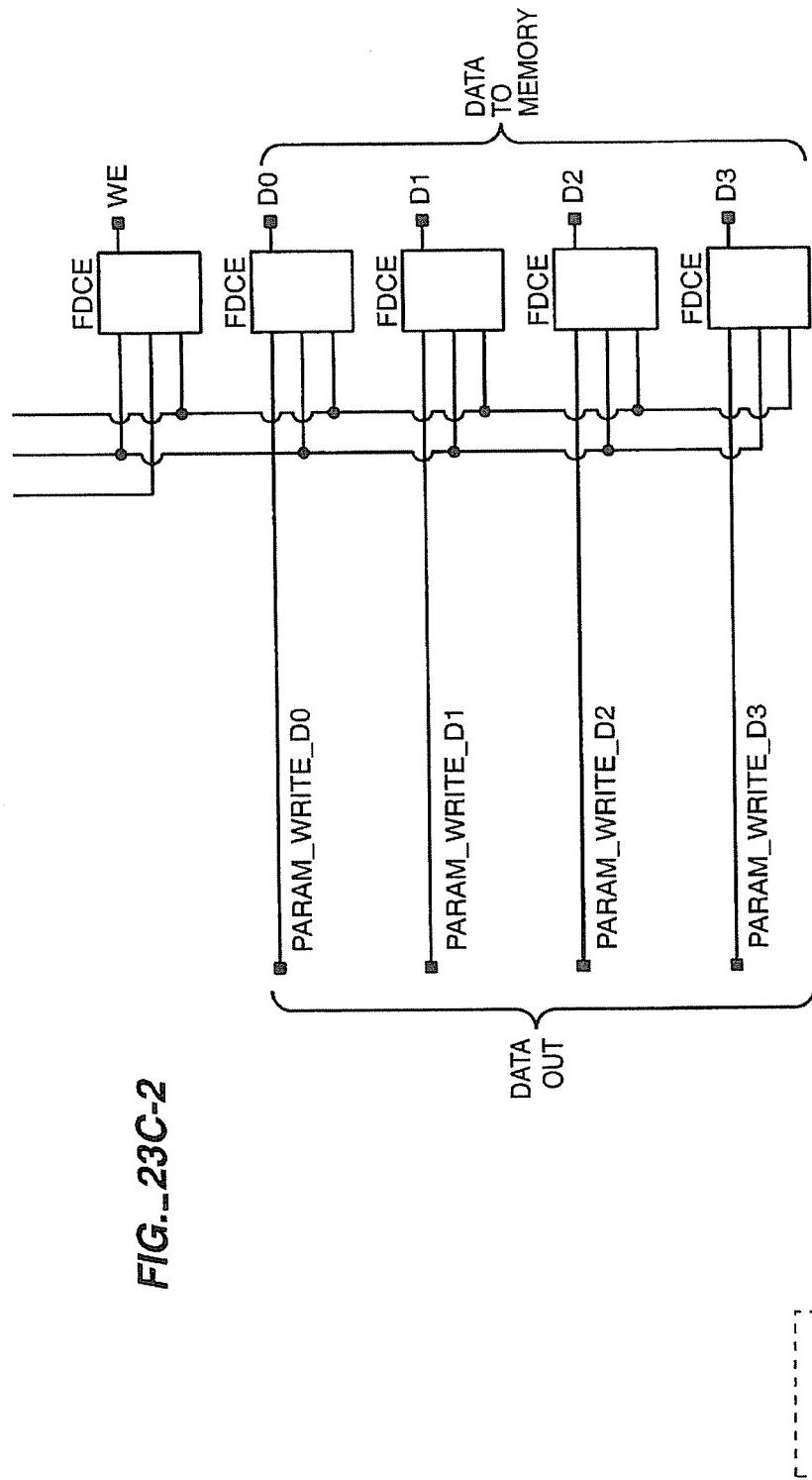


FIG. 23C-1

FIG. 23C-2

FIG. 23C

U.S. Patent May 1, 2001 Sheet 79 of 90 US 6,226,776 B1

```
typedef struct
{
    int a;
    int b;
} STRUCTURE;

int read_member_b (STRUCTURE *p)
{
    return p -> b;
}
```

FIG._24A

```
module Struct1(clock,reset,a,d,o,we,result_read_member_b,param_read_member_b_p,
              run_read_member_b,ready_read_member_b);
    input      clock, reset, run_read_member_b;
    input [ 3:0] o, param_read_member_b_p;
    output     we, ready_read_member_b;
    output [ 3:0] a, d, result_read_member_b;
    reg        we, ready_read_member_b, state;
    reg [ 3:0] a, d, result_read_member_b, _1_4;

    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
                0:begin
                    if (run_read_member_b)
                    begin
                        ready_read_member_b = 0;
                        a = (param_read_member_b_p + 4'd1);
                        state = 1;
                    end
                end
                1:begin
                    _1_4 = o;
                    result_read_member_b = _1_4;
                    ready_read_member_b = 1;
                    state = 0;
                end
                default: ;
            endcase
        end
    end
endmodule
```

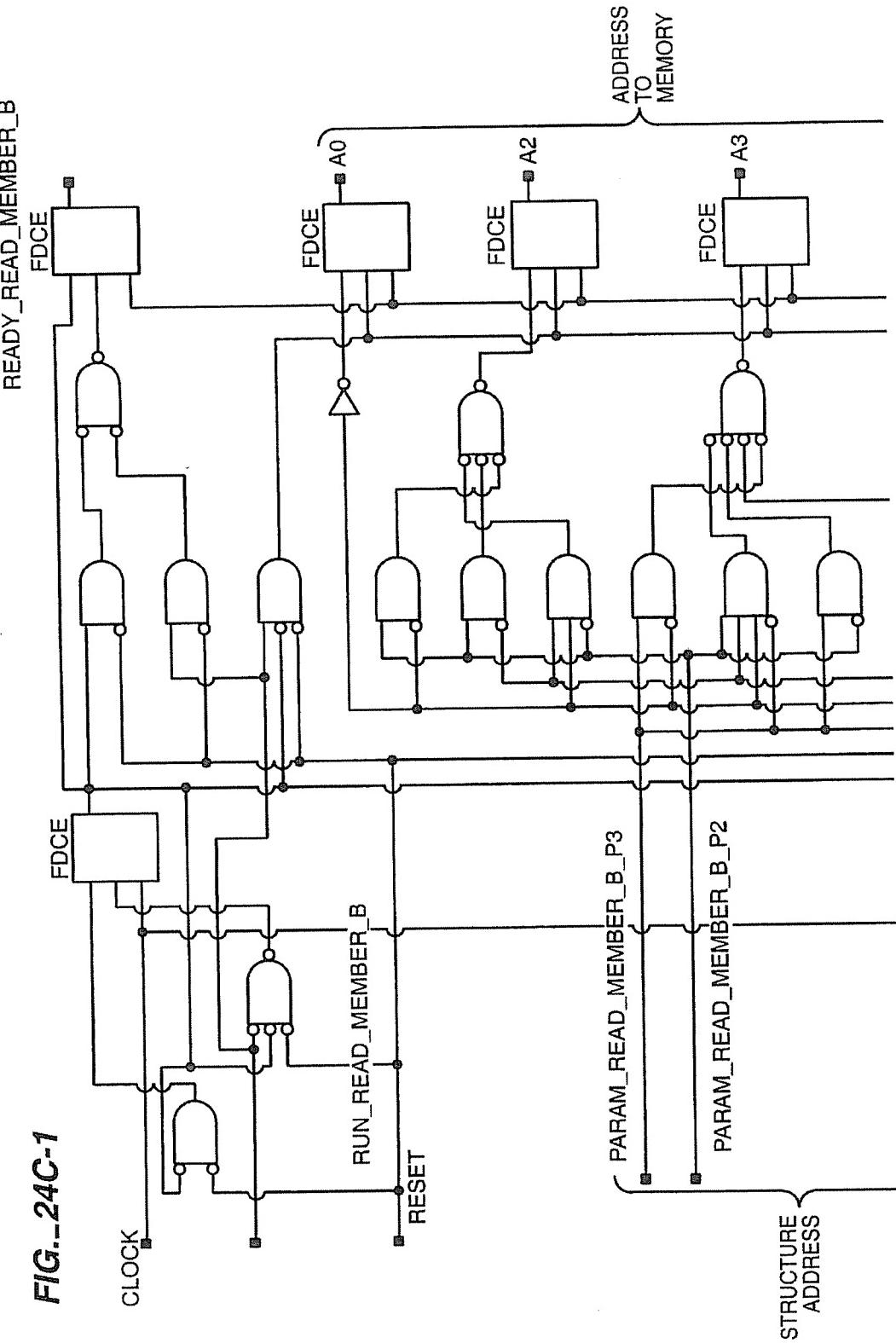
FIG._24B

U.S. Patent

May 1, 2001

Sheet 80 of 90

US 6,226,776 B1

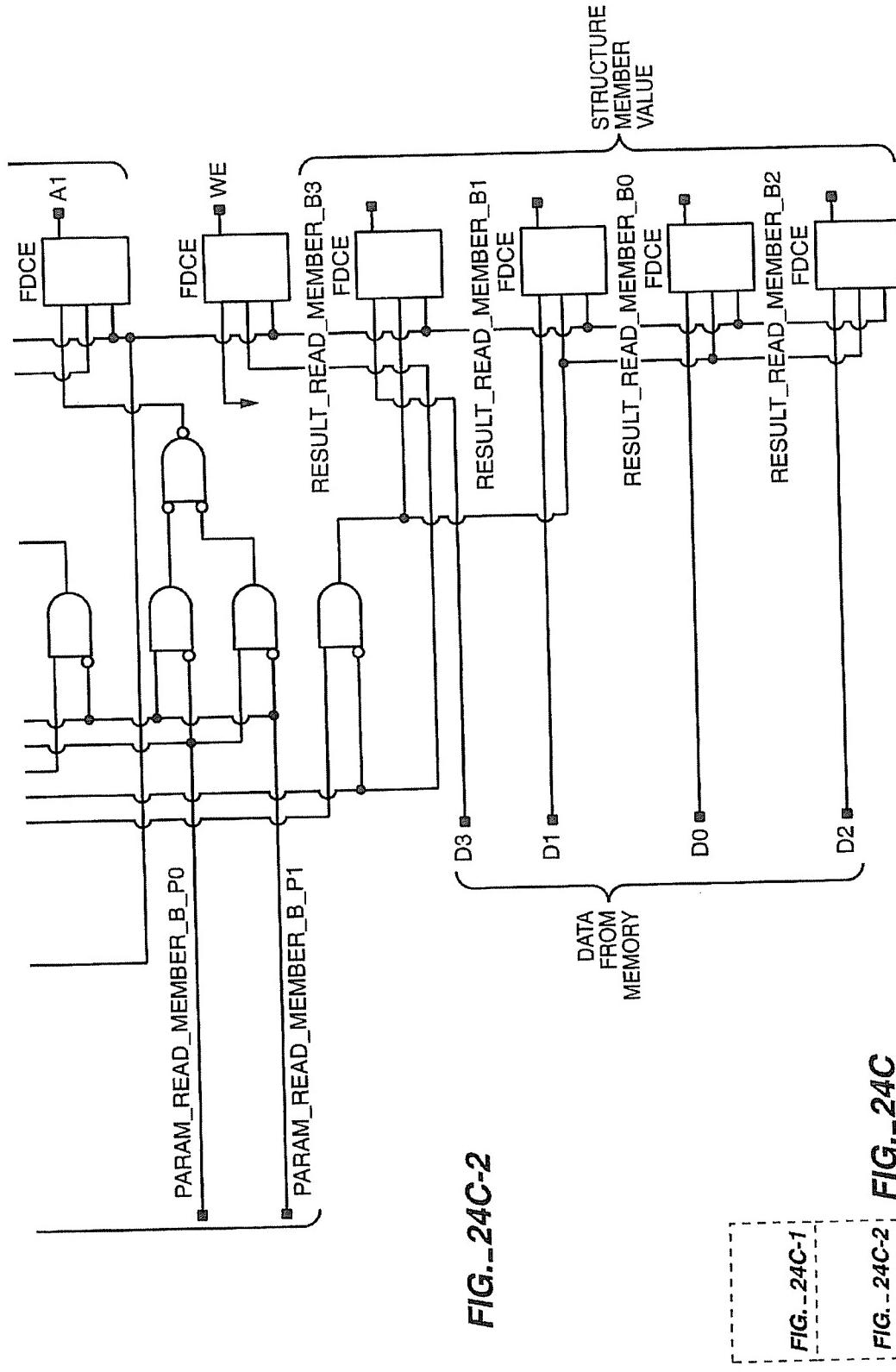


U.S. Patent

May 1, 2001

Sheet 81 of 90

US 6,226,776 B1



U.S. Patent May 1, 2001 Sheet 82 of 90 US 6,226,776 B1

```

typedef struct
{
    int a;
    int b;
} STRUCTURE;

void write_member_b (STRUCTURE *p, int n)
{
    p -> b = n;
}

```

FIG._25A

```

module Struct2(clock,reset,a,d,o,we,param_write_member_b_p,param_write_member_b_n,
               run_write_member_b,ready_write_member_b);
    input      clock, reset, run_write_member_b;
    input [ 3:0] o, param_write_member_b_p, param_write_member_b_n;
    output     we, ready_write_member_b;
    output [ 3:0] a, d;
    reg        we, ready_write_member_b, ready_write_member_b, state;
    reg [ 3:0] a, d;

    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
                0:begin
                    if (run_write_member_b)
                    begin
                        ready_write_member_b = 0;
                        a = (param_write_member_b_p + 4'd1);
                        d = param_write_member_b_n;
                        we = 1;
                        state = 1;
                    end
                end
                1:begin
                    we = 0;
                    ready_write_member_b = 1;
                    state = 0;
                end
                default: ;
            endcase
        end
    end
endmodule

```

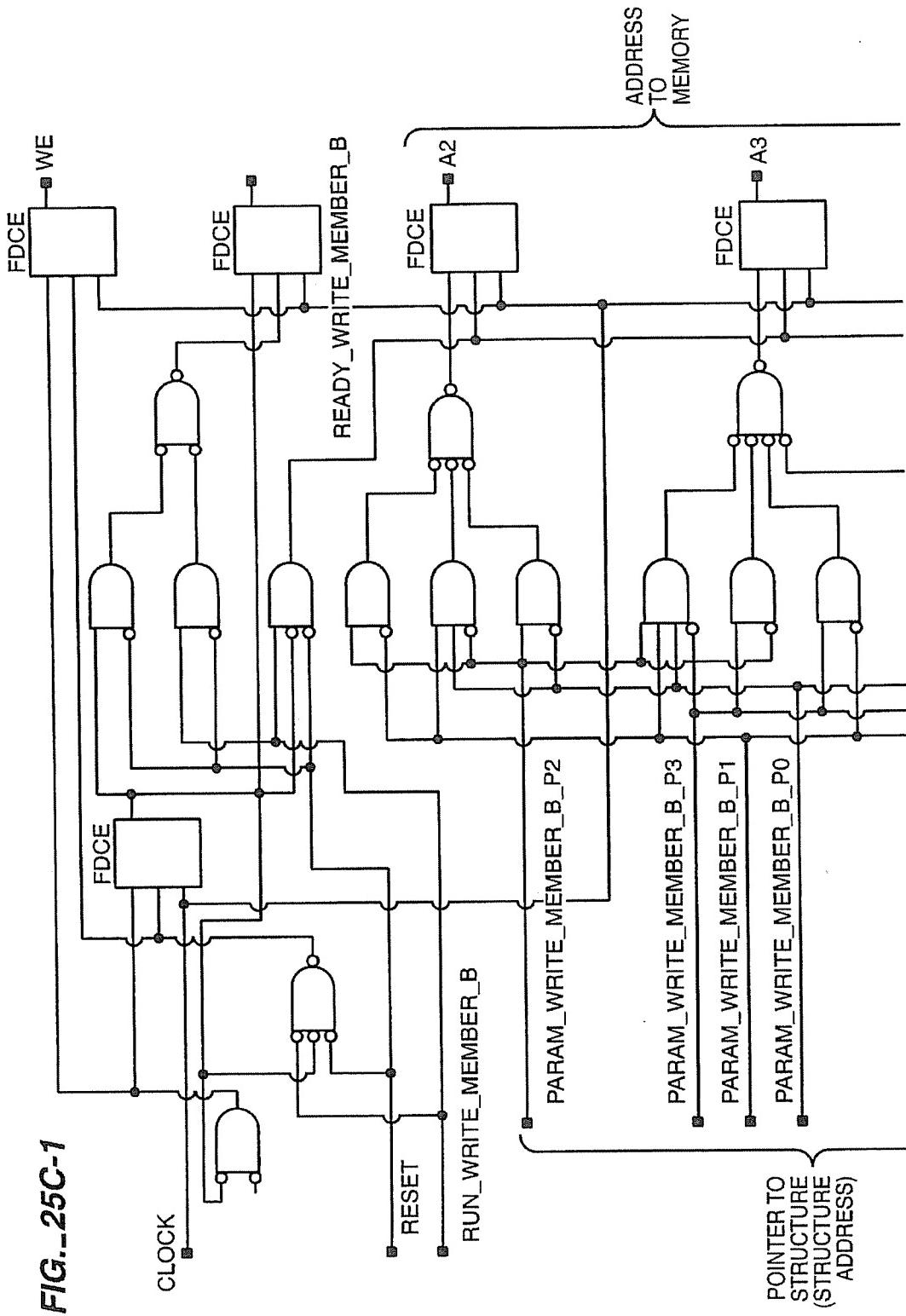
FIG._25B

U.S. Patent

May 1, 2001

Sheet 83 of 90

US 6,226,776 B1



U.S. Patent

May 1, 2001

Sheet 84 of 90

US 6,226,776 B1

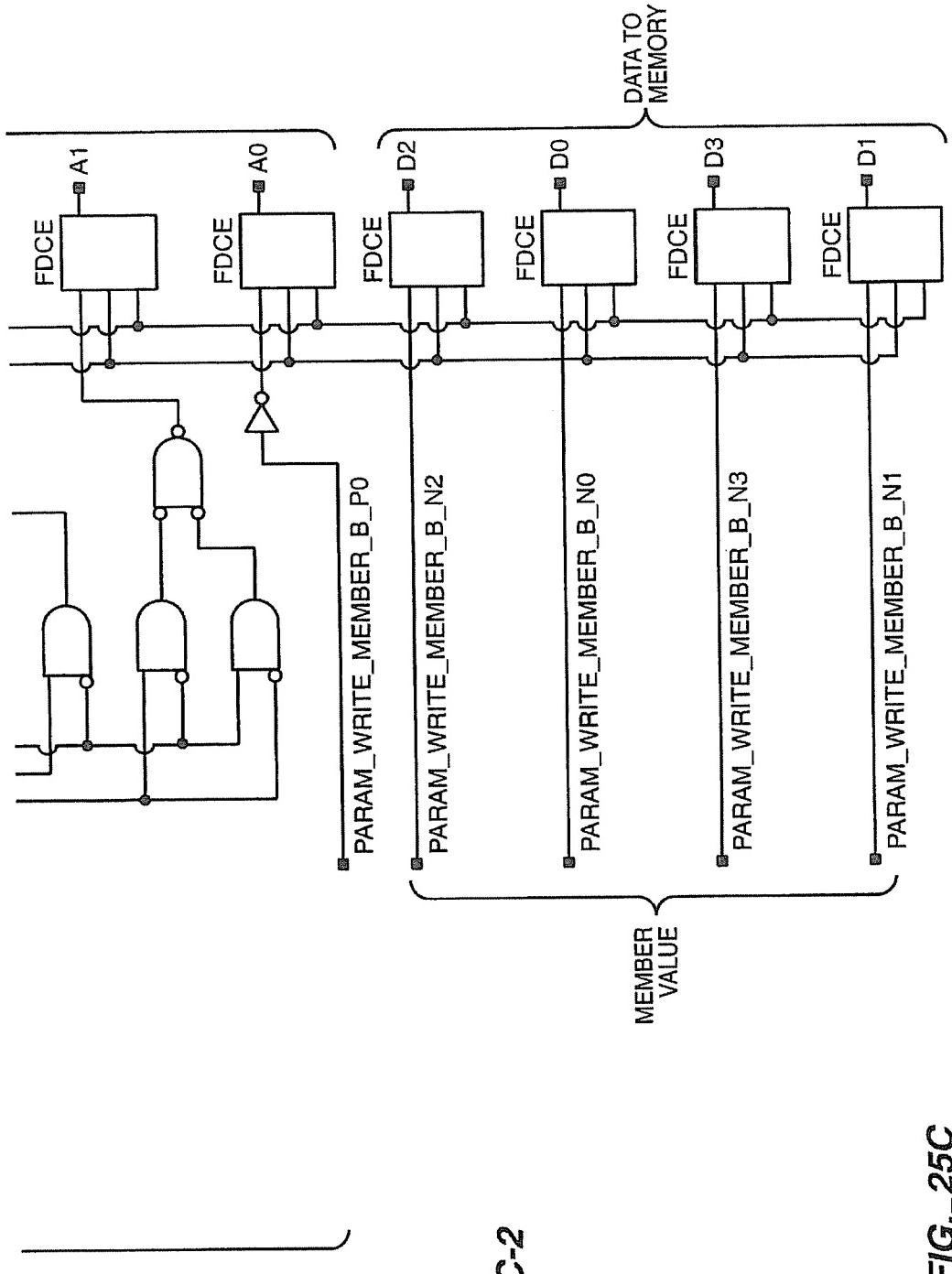


FIG.-25C-2

FIG.-25C-1

FIG. - 25C-2

U.S. Patent

May 1, 2001

Sheet 85 of 90

US 6,226,776 B1

```

typedef struct
{
    int a;
    int b;
} STRUCTURE;
STRUCTURE s;
int read_member_b (void)
{
    return s.b;
}

```

FIG._26A

```

`define _s 2'h1
module Struct4(clock,reset,a,d,o,we,result_read_member_b,run_read_member_b,ready_read_member_b);
    input      clock, reset, run_read_member_b;
    input [ 3:0] o;
    output     we, ready_read_member_b;
    output [ 1:0] a, d, result_read_member_b;
    reg        we, ready_read_member_b, state;
    reg [ 1:0]  a, d, result_read_member_b;
    reg [ 3:0]  _1_4 ;
    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
                0: begin
                    if (run_read_member_b)
                    begin
                        ready_read_member_b = 0;
                        a = (_s + 4'd1);
                        state = 1;
                    end
                end
                end
                1: begin
                    _1_4 = o;
                    result_read_member_b = _1_4;
                    ready_read_member_b = 1;
                    state = 0;
                end
                default: ;
            endcase
        end
    end
endmodule

```

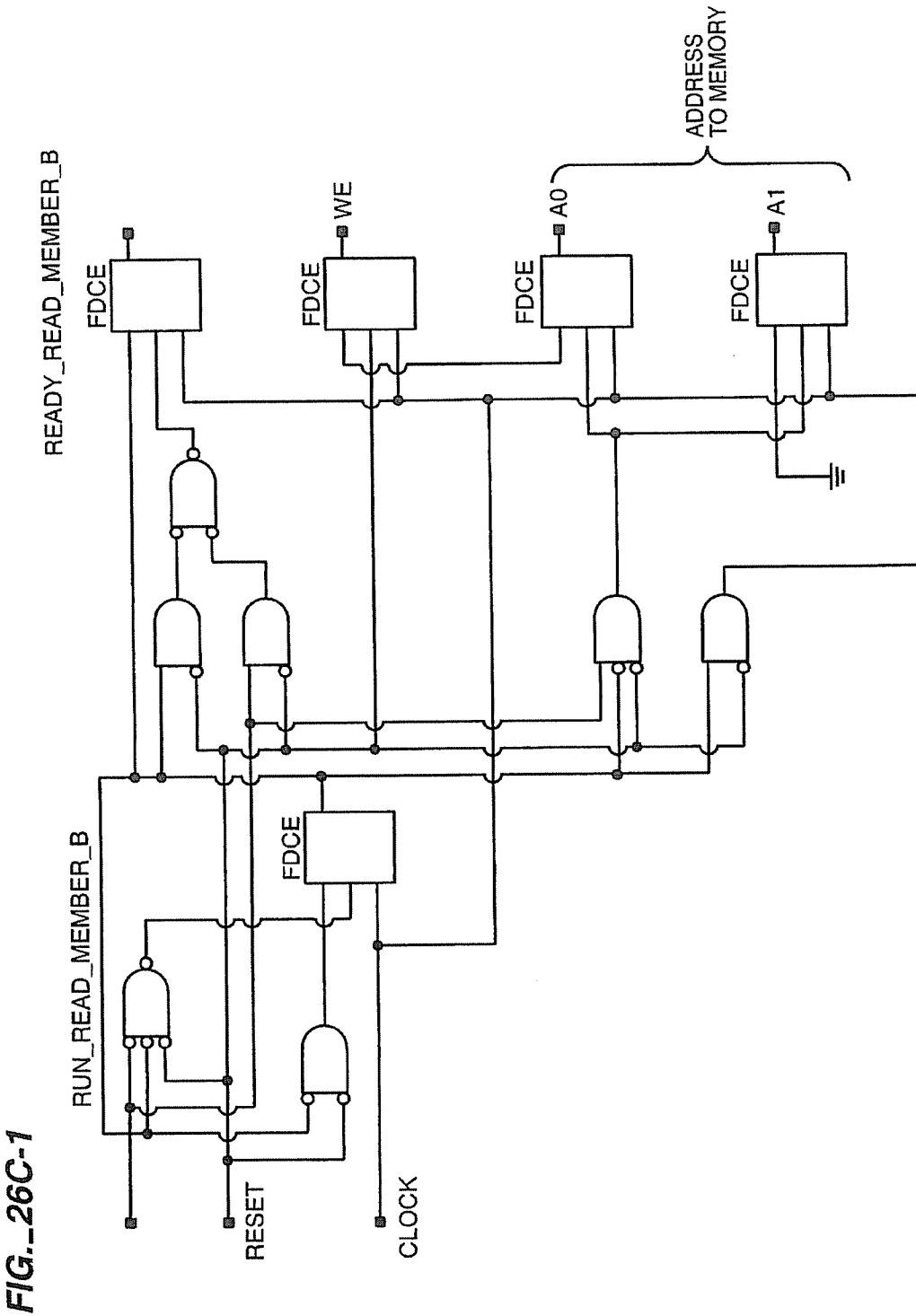
FIG._26B

U.S. Patent

May 1, 2001

Sheet 86 of 90

US 6,226,776 B1



U.S. Patent

May 1, 2001

Sheet 87 of 90

US 6,226,776 B1

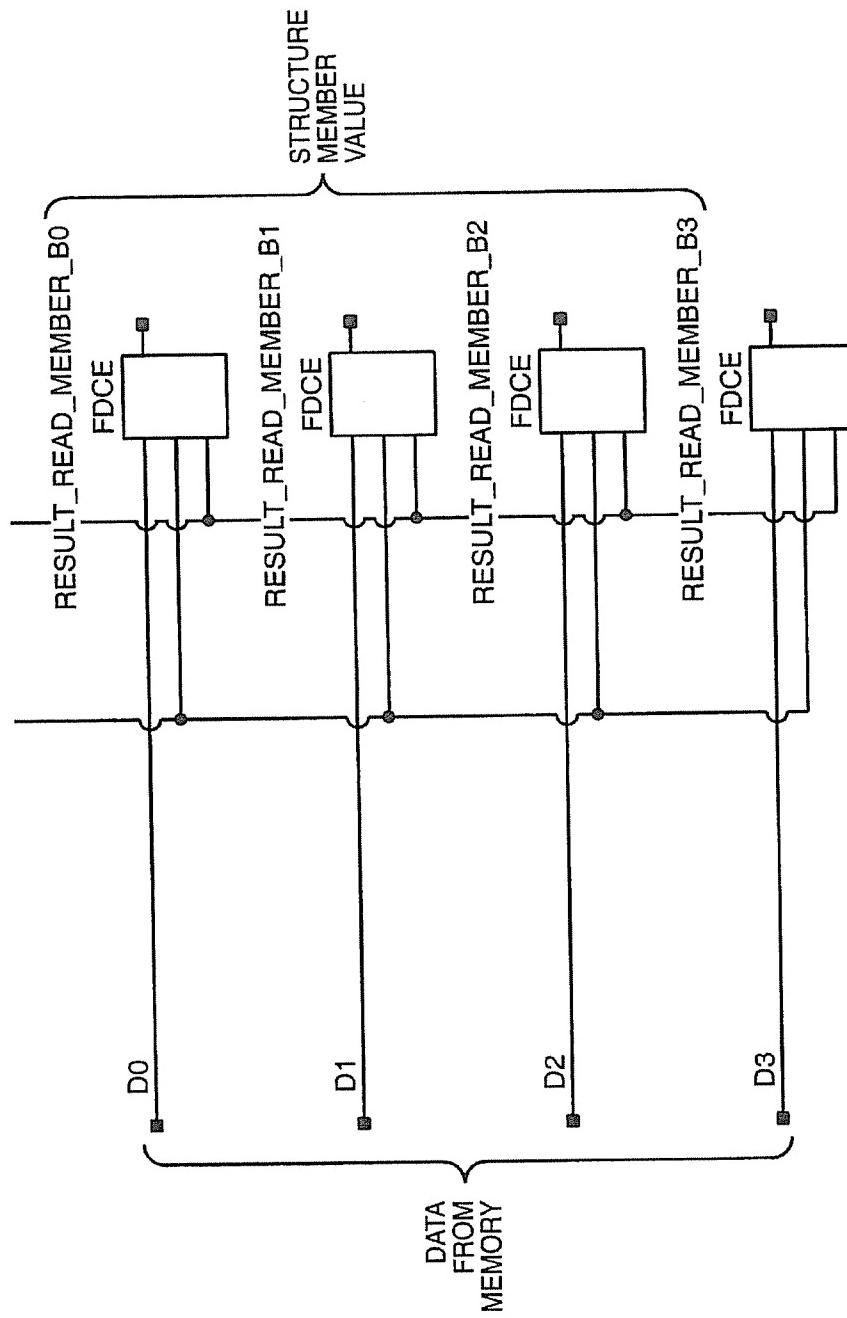


FIG. 26C-2



FIG. 26C-1

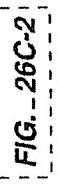


FIG. 26C-2

FIG. 26C

U.S. Patent May 1, 2001 Sheet 88 of 90 US 6,226,776 B1

```
typedef struct
{
    int a;
    int b;
} STRUCTURE;
STRUCTURE s;
void write_member_b (int n)
{
    s.b = n;
}
```

FIG._27A

```
`define _s 2'h1
module Struct5(clock,reset,a,d,o,we,param_write_member_b_n,run_write_member_b,ready_write_member_b);
    input      clock, reset, run_write_member_b;
    input [ 3:0] param_write_member_b_n, o;
    output     we, ready_write_member_b;
    output [ 1:0] a;
    output [ 3:0] d;
    reg        we, ready_write_member_b, state;
    reg [ 1:0] a;
    reg [ 3:0] d;
    always @(posedge clock)
    begin
        if (reset)
        begin
            we = 0;
            state = 0;
        end
        else
        begin
            case (state)
                0: begin
                    if (run_write_member_b)
                    begin
                        ready_write_member_b = 0;
                        a = (_s + 4'd1);
                        d = param_write_member_b_n;
                        we = 1;
                        state = 1;
                    end
                end
                1: begin
                    we = 0;
                    ready_write_member_b = 1;
                    state = 0;
                end
            default: ;
        endcase
        end
    end
endmodule
```

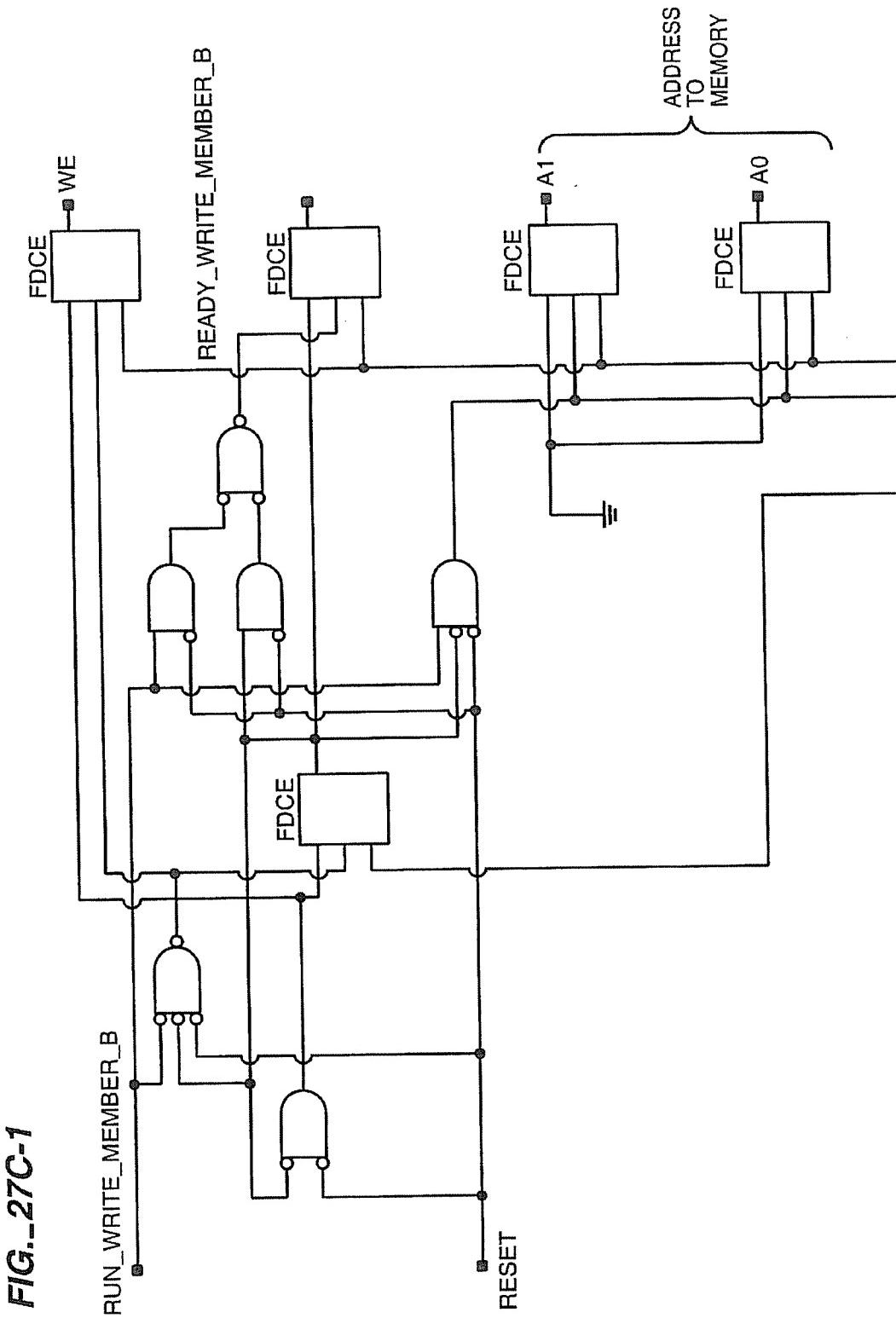
FIG._27B

U.S. Patent

May 1, 2001

Sheet 89 of 90

US 6,226,776 B1



U.S. Patent

May 1, 2001

Sheet 90 of 90

US 6,226,776 B1

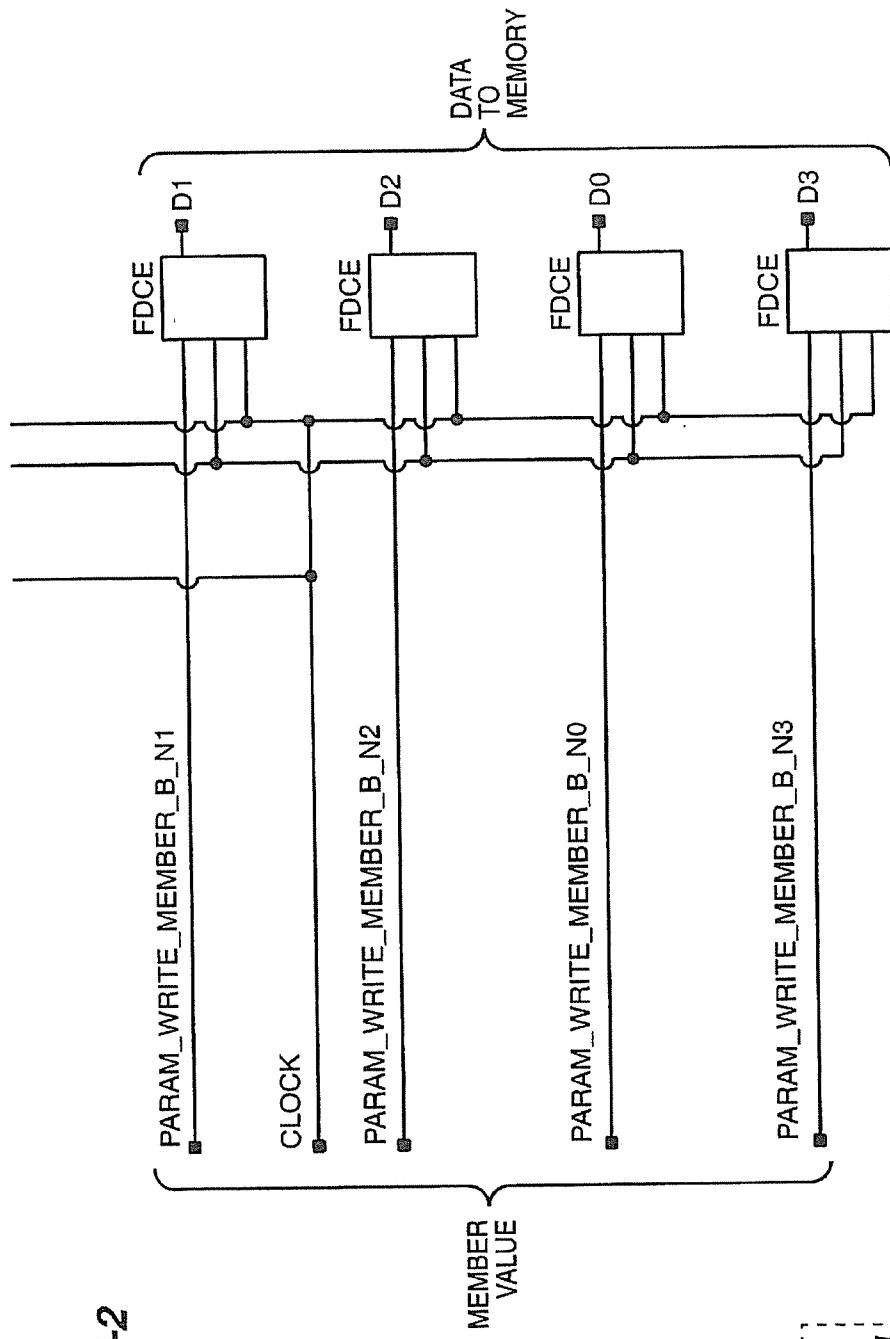


FIG. 27C-2

FIG. 27C-1

FIG. 27C-2

FIG. 27C

US 6,226,776 B1

1

SYSTEM FOR CONVERTING HARDWARE DESIGNS IN HIGH-LEVEL PROGRAMMING LANGUAGE TO HARDWARE IMPLEMENTATIONS

CROSS-REFERENCE TO MICROFICHE APPENDIX

Appendix A, which is a part of the present disclosure, is a microfiche appendix consisting of two sheets of microfiche having a total of 176 frames. Microfiche Appendix A is a source code listing of a portion of the code comprising one embodiment of a system for converting hardware designs in a highlevel programming language (ANSI C) into a register transfer level hardware description language (Verilog), which is described in more detail below.

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as they appear in the U.S. Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

FIELD OF THE INVENTION

This invention relates to configuring digital circuits, especially computationally intensive digital circuitry, such as field programmable gate arrays (FPGAs) and other programmable logic hardware and application specific integrated circuits (ASICs), and, more particularly, to computer aided design of such computationally intensive digital circuitry. Specifically, one embodiment of the invention provides a system for converting a hardware design rendered in a high-level programming language, such as ANSI C, into an actual hardware implementation, for example, as an FPGA.

Various persons have previously attempted to address the long-felt need for an easy approach to designing computationally intensive digital circuits, such as FPGAs. The prior art evidences two divergent approaches for designing actual implementations for digital hardware.

One prior art approach has been to create a specific hardware description language (HDL) for designing hardware. Various commercially available HDLs have been developed, such as Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. After a hardware design is rendered in an HDL, the HDL design is processed into a gate-level hardware representation using one of various hardware- or manufacturer-specific synthesis programs to interpret the HDL design. Then, the gate-level representation is reduced to an actual hardware implementation using conventional physical design tools.

HDLs are specific computer aided design tools for hardware designers and require a level of expertise in the use of the particular HDL being employed to render a hardware design and are difficult to learn and use. See, for example, Tuck, B., "Raise Your Sights to the System Level," Computer Design, June, 1997, pp. 53, 69. Therefore, only persons who frequently design hardware typically use HDLs. If a circuit application arises in which design and implementation of an FPGA is economically justified, the vast majority of persons must retain an expert or confront the difficult task of learning an HDL. Furthermore, these HDLs are not typically universal for the design of FPGAs, since many HDLs are supported by only a single or a limited number of hardware manufacturers. Consequently, even experienced users of an HDL, such as full-time hardware designers, may not be sufficiently familiar with other HDLs to be able to

2

render a design which can be implemented in a variety of hardware systems from a number of different hardware manufacturers. There is therefore a need for an easy-to-use, universal computer aided hardware design tool by both experts and occasional hardware designers as well.

A second prior art approach recognizes that the development of computationally intensive hardware would be available to a wider population of persons if hardware designs could be rendered in a standard high-level programming language which is more universally known, easier to use, and more frequently employed than HDLs. One such high-level programming language is ANSI C. Others, many of which have several extensions, are: APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forte, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

Unfortunately, prior art attempts to base computer aided hardware design tools on high-level programming languages have been incomplete. Consequently, these attempts to realize the goal of providing an effective tool for designing hardware using a high-level programming language have not achieved widespread use. All the prior art attempts have each suffered three or more of the following shortcomings:

1. Prior art computer aided hardware design tools generate so many unhandled exceptions and language restrictions when presented with typical high-level language programs (e.g., a C program) that it renders such tools virtually useless for practical hardware design. Therefore, persons using such tools are required to read manuals and attempt to create work-arounds for portions of the high-level programming language, which are not implemented.

2. Hardware generated using prior art computer aided hardware design tools is typically based on simple line-by-line, table-driven translation and as such does not contain the necessary widely scoped optimizations that produce practical and useful hardware designs for most applications.

3. Design output of the prior art computer aided hardware design tool can be in so-called "behavioral HDL" for modeling hardware but in many cases cannot be synthesized by existing synthesis programs into a gate-level representation of the hardware.

4. Prior art computer aided hardware design tools attempt to generate designs which only apply to a single manufacturer hardware family, such as the creation of an XNF file for the Xilinx XC4000 FPGA family.

5. The language employed by prior art computer aided hardware design tools has added and limited so many constructs that it is difficult to consider the employed language as the high-level programming language purportedly being emulated. That is, the result language is more like a new specialized HDL.

Considered in more detail, one prior art computer aided hardware design tool which purports to enable persons to render hardware designs in a high-level programming language is Transmogrifier (TMCC), a tool developed in 1994-1995 at the University of Toronto, Canada. A comprehensive compiler typically requires at least ten times the 4,900 lines of code which comprise TMCC. TMCC does not support such common C-type programming language features as "do" loops, "for" loops, local or global arrays (i.e., the [] operator), the operators ->, unary &, unary *, etc.

Another prior art computer aided hardware design tool is NLC developed at Ecole Polytechnique Federale de

US 6,226,776 B1

3

Lausanne, Switzerland in 1994–1995. The code for NLC is based on the public domain compiler known as GNU. Like Transmogrifier, NLC supports a severely restricted subset of C-type programming language, for example, “for” loop bounds must be known at compile time, and functions cannot return values. The NLC library generates hardware designs only in XNF format for the Xilinx XC4000 FPGA family.

A prior art computer aided design tool for hardware which purports to be a high-level programming language is Handel-C, originally developed at the Computing Laboratory of Oxford University, England in 1995–1996 and presently supported by Embedded Solutions Limited in Berkshire, England. Although Handel-C has been described by its developers as a subset of ANSI C with some Occam-like extensions, the vast majority of ANSI C programs cannot be processed by Handel-C. For example, functions in Handel-C must be declared in main, have no return values, and have no parameters. There are no structs, unions, enums, struct bit fields, character constants, externs, pointers, etc. The output of Handel-C is in XNF format for the Xilinx XC4000 FPGA family. Since the language is so unlike a high-level programming language, Handel-C is better classified as an HDL for the xilinx XC4000 FPGA family.

A further prior art computer aided hardware design tool is XC. XC was developed at Giga Operations Corporation in Berkeley, Calif. XC appears to be generally disclosed in Taylor, U.S. Pat. No. 5,535,342 issued on Jul. 9, 1996 and in Taylor et al., U.S. Pat. No. 5,603,043 issued on Feb. 11, 1997, both assigned to Giga Operations. There is little resemblance to C-type programming language. XC does not appear to implement structures, pointers, multidimensional arrays, unary &, unary *, etc., as evidenced by various examples published by Giga Operations at its web site. The output of XC is targeted only at the XC4000 board manufactured by Giga Operations that incorporates the Xilinx XC4000 FPGA family.

Finally, an analysis of the prior art would be incomplete without mention of the JRS ANSI C to VHDL Translator (JRS) developed at JRS Research Laboratories, Inc., Orange, Calif. in 1996. JRS appears to handle a wider range of high-level programming language features than other known prior art attempts but still lacks such standard features as multidimensional arrays, abstract declarators, automatic type conversions, pointers to arrays, pointers to functions, functions returning pointers, variable-length argument lists, and “goto” statements. An even more significant deficiency of JRS is that the output of JRS is in general not synthesizable VHDL. JRS utilizes constructs in VHDL considered “behavioral VHDL,” that are useful to describe hardware for simulation, but in general cannot be implemented into gate-level hardware representations by existing synthesis tools. As such, JRS is appropriately classified as a language translator, not a compiler that outputs realizable hardware designs. JRS does not address register transfer level (RTL) HDL problems of translating a high-level language to a hardware design.

Consequently, there is a long-felt but unsolved need for a system in which a hardware design can be rendered in a high-level programming language in a way to enable the full power of that language to be employed as a computer aided design tool for hardware. There is also a concomitant need for a computer aided design tool for hardware which is familiar and therefore easier to use than HDLs and which is nevertheless universal in that the resulting hardware designs can be implemented by a broad range of hardware manufacturers.

4

SUMMARY OF THE INVENTION

The present invention provides a solution to the problem of facilitating the design of an actual hardware implementation for digital circuits using a high-level programming language by providing in one embodiment a system for converting a C-type language program to a hardware design implementation, such as an FPGA or other programmable logic or an ASIC. A method in accordance with one embodiment of the invention comprises the steps of creating an algorithmic representation for a preliminary hardware design in a given C-type programming language and compiling the C-type programming language description into a register transfer level (RTL) hardware description language (HDL) design which can be synthesized. Preferably, the method of the invention additionally comprises the step of synthesizing the HDL design into a gate-level hardware representation using a conventional synthesis program to interpret the HDL design.

In order to produce an actual hardware implementation, the method in accordance with the invention preferably further comprises the step of using conventional physical design tools to implement the gate-level hardware representation as an actual hardware implementation. These physical design tools include a place and route program. In the case of an FPGA, the physical design tools also include a hardware configuration system which processes the file produced by the place and route program and creates a bit stream to personalize the FPGA. In the case of an ASIC, the physical design tools include a place and route program for creating a file for patterning an integrated circuit mask for use in fabricating the ASIC.

The C-type programming language used to render an algorithmic representation for the preliminary hardware design may be any standard C-type programming language including ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C. The method of the invention is robust in that the method converts any typical hardware design rendered with the C-type programming language into a standard HDL. The standard HDLs include Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. The method in accordance with the invention first converts the preliminary hardware design to a register transfer level HDL, which enables the HDL design to be synthesized into a gate-level hardware representation using a standard synthesis program and then implemented as an actual hardware implementation using conventional physical design tools. In accordance with the invention, an embodiment of apparatus comprising a personal computer or workstation and executable code that implements the method of the invention is also provided.

The method in accordance with the invention can further comprise simulating the HDL synthesizable design prior to synthesizing the HDL design. Any conventional simulation program used to simulate a standard HDL design can be employed for simulating the hardware design converted from a rendition in C-type programming language to HDL. This enables the design originally rendered in a C-type programming language to be simulated and the design to be modified before the hardware is actually synthesized and implemented using conventional physical design tools. Consequently, hardware designs rendered in the C-type programming language can be optimized before the hardware implementation phase. Unlike the prior art, the method of the invention enables a hardware design originally rendered in a C-type programming language to be both simulated and implemented in an actual hardware implementation. The apparatus in accordance with the invention can

US 6,226,776 B1

5

also be utilized to simulate the HDL synthesizable design following compilation to HDL using code that executes the standard simulation program.

In a preferred embodiment of the method in accordance with the invention, compilation of the C-type programming language into HDL comprises mapping predetermined C-type programming language expressions to functionally equivalent HDL program language expressions, assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design, and configuring in the HDL synthesizable design an interface for the gate-level hardware representation. This is sufficient for basic hardware designs rendered in the C-type programming language.

Additionally, efficient conversion of basic hardware designs from the C-type programming language to HDL preferably entails compilation of a plurality of selected C-type functions that can execute simultaneously into a plurality of executable HDL program language expressions that operate in parallel. Efficient conversion further entails compilation of a plurality of interdependent C-type functions into a plurality of executable HDL program language expressions that operate either simultaneously or sequentially in a data processing pipeline. Preferably, in order to effect efficient conversion, a callable C-type function is also compiled into an executable HDL program language expression that is executed either synchronously or a synchronously on the occurrence of an external event.

In the preferred embodiment of the method in accordance with the invention, compilation can comprise compiling a C-type program control flow into an HDL state machine. The method further comprises assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design and configuring in the HDL synthesizable design an interface for the gate-level hardware representation.

Also, the C-type program is preferably analyzed for determining the presence of any C-type pointers in the C-type program and compiling any C-type pointers and pointer indirection into an HDL state-machine-based memory access protocol. Therefore, unlike prior art computer aided hardware design tools, the method in accordance with the invention supports the use of memory structures and pointers typically employed in C-type programming language during the original rendering of the hardware design.

Additionally, the C-type program is preferably analyzed for determining the presence of any non-addressable variables in the C-type program and mapping any non-addressable variables onto hardware registers formed in HDL. Similarly, the C-type program is also analyzed for determining the presence of any addressable variables in the C-type program and mapping any addressable variables onto addressable hardware memory and/or an addressable array of hardware registers formed in HDL.

Finally, the C-type program is preferably analyzed for determining the presence of any complicated C-type mathematical operations in the C-type program and compiling multiple occurrences of any complicated C-type mathematical operation into a given HDL functional block that implements such an operation invoked on each occurrence by different states driven by the state machine. The complicated C-type mathematical operations include integer operations, such as signed and unsigned multiplication, signed and unsigned division, signed and unsigned remainder operation, variable-shift left shift, variable-shift right shift with or without sign extension, etc., and floating point

6

operations, such as addition, subtraction, subtraction from zero (unary-), multiplication, division, and conversions involving floating point types. The given HDL functional block is effectively re-used to provide resource-sharing for conversion to an efficient HDL synthesizable design.

Preferably, compilation in accordance with the method of the invention employs state machines for efficient conversion from the C-type programming language to the HDL program language. More particularly, a plurality of selected C-type functions that can execute simultaneously are compiled into a plurality of HDL state machines that operate in parallel. A plurality of selected C-type functions that can execute simultaneously are further preferably compiled into a plurality of HDL state machines that operate in parallel. Also, a plurality of interdependent C-type functions are compiled into a plurality of executable HDL program language expressions that operate either simultaneously or sequentially in a data processing pipeline. A callable C-type function is also preferably compiled into at least one executable HDL state machine which is executed either synchronously or a synchronously on the occurrence of an external event.

Also in accordance with the present invention, a solution is provided to the problem of facilitating the design of an actual hardware implementation using any high-level programming language by providing in one embodiment a system for initially translating from an algorithmic representation for a hardware design in any given standard high-level programming language to an algorithmic representation for the hardware design in a C-type programming language and then converting the C-type language program to a hardware design implementation, such as an FPGA or other programmable logic or an ASIC. Standard high-level programming languages that can be translated to a C-type programming language utilizing conventional compilers include APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

Accordingly, the invention also provides in one embodiment a method which comprises the steps of creating an algorithmic representation for a preliminary hardware design in a given high-level programming language, translating the high-level programming language preliminary hardware design into an algorithmic representation for the preliminary hardware design in a C-type programming language, and compiling the C-type programming language description into a register transfer level HDL design which can be synthesized. Preferably, the method of the invention additionally comprises the step of synthesizing the HDL design into a gate-level hardware representation using a conventional synthesis program to interpret the HDL design. In order to produce an actual hardware implementation, the method in accordance with the invention preferably further comprises the step of using conventional physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

The steps of compiling the C-type programming language description for the preliminary hardware design into an HDL synthesizable design and synthesizing the HDL design using a conventional synthesis program are identical to the steps performed if the hardware design were initially rendered in the C-type programming language. Also, the step of then implementing the gate-level representation resulting from

US 6,226,776 B1

7

synthesizing the HDL design as an actual hardware implementation using conventional physical design tools is identical to the step performed after synthesis in the case where the algorithmic representation for the hardware design was initially rendered in the C-type programming language. This enables hardware designs rendered in virtually any standard high-level programming language to be implemented in hardware. Also, the method can further comprise simulating the HDL synthesizable design prior to synthesizing the HDL design. In accordance with the invention, an embodiment of apparatus comprising a personal computer or workstation and executable code that implements the method of the invention is also provided.

The system in accordance with the invention has significant advantages compared to the purported high-level language computer aided hardware design tools known in the prior art. The system of the invention is the only ANSI C compiler. The system in accordance with the invention supports C-type programming language pointers and pointer arithmetic, variable argument functions, pointers to functions, recursion, structure assignment, and floating point arithmetic. The system of the invention also supports C-type programming language structures, unions, enums, and typedefs, as well as the complete set of C-type programming language control flow operators, including "for," "while", and "do" loops, "if", "switch", "break", "continue", "goto", and "return". The system in accordance with the invention also supports C char type, including strings literals, character constants, character arrays, and character pointers. All basic C integer types and type conversions are supported by the system of the invention, and the system follows ANSI C rules for integer promotion in operations. The system in accordance with the invention converts an ANSI C algorithmic representation for a hardware design to a synthesizable hardware design in HDL, such as Verilog, and, therefore, is compliant with multiple hardware technologies, whereas the known prior art computer aided hardware design tools generate hardware- and/or manufacturer-specific files, such as an XNF file for the Xilinx XC4000 FPGA family. Unlike prior art computer aided hardware design tools that generate "behavioral VHDL" models that can only be simulated, the system of the invention enables both simulation and synthesis that leads to an actual hardware implementation.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objectives and features and the concomitant advantages of the present invention will be better understood and appreciated by those skilled in the art in view of the description of the preferred embodiments given below in conjunction with the accompanying drawings. In the drawings:

FIG. 1 illustrates one embodiment of the computer aided hardware design tool for compiling C-type programming language into hardware description language (HDL) to enable conversion from a hardware design in a high-level programming language to a synthesizable HDL hardware design in accordance with the invention;

FIG. 2 is a flowchart of one embodiment of the method in accordance with the invention for compiling high-level programming language, such as C-type programming language, into HDL to enable conversion from a hardware design in a high-level programming language to a synthesizable HDL hardware design;

The remainder of the figures illustrates various examples of compilation of C-type programming language into a register transfer level (RTL) HDL to enable conversion from

8

a hardware design in a high-level programming language to a synthesizable HDL hardware design by the computer aided hardware design tool in accordance with the invention shown in FIG. 1, specifically:

5 FIG. 3A shows an initial C-language description having simple expression functions ("func1" and "func2");

FIG. 3B shows HDL created by compiling the C-type program shown in FIG. 3A, illustrating simple expression functions ("func1" and "func2") which do not require state machines implemented in RTL Verilog;

10 FIG. 4A shows an initial C-language description having "non-simple" expression functions ("func1", "func2", and "func3") using data conversions and "if-then-else" statements;

15 FIG. 4B shows HDL created by compiling "non-simple" expression functions ("func1", "func2", and "func3") of the C-type program shown in FIG. 4A, which do not require state machines implemented in RTL Verilog;

20 FIG. 5A shows an initial C-language description using "for" loop expressions and non-addressable variables;

FIG. 5B, comprising FIGS. 5B1-5B4, shows HDL created by compiling the C-type program shown in FIG. 5A, including a "for" loop and non-addressable variables 25 mapped into registers using a state machine implemented in RTL Verilog;

FIG. 6A shows an initial C-language description with input/output (I/O) function calls;

30 FIG. 6B shows HDL created by compiling ANSI C printf and scanf arguments of the C-type program shown in FIG. 6A into wire assignments in RTL Verilog;

FIG. 7A shows an initial C-language description with multiple occurrences of complicated mathematical operations;

35 FIG. 7B, comprising FIGS. 7B1-7B4, shows HDL created by compiling the C-type program shown in FIG. 7A into functional blocks in RTL Verilog for complicated mathematical operations invoked on each occurrence by different states of the generated state machine;

FIG. 8A shows an initial C-language description having interdependent C-type functions;

40 FIG. 8B shows HDL created by compiling the C-type program shown in FIG. 8A into a simple data processing pipeline implemented in RTL Verilog;

FIG. 8C is a block diagram of the data processing pipeline implemented in the RTL Verilog shown in FIG. 8B;

FIG. 9A shows an initial C-language description having floating point arithmetic;

45 FIG. 9B, comprising FIGS. 9B1-9B4, shows HDL created by compiling the C-type program shown in FIG. 9A, illustrating subtraction from zero (unary-), addition, multiplication, subtraction, division, and conversion operations involving floating point arithmetic in RTL Verilog;

FIG. 10A shows an initial C-language description of two complex number addition with structures, including structure assignments, structure function parameters, and structure function return values;

50 FIG. 10B, comprising FIGS. 10B1-10B5, shows HDL created by compiling the C-type program shown in FIG. 10A, illustrating structure function parameters and structure return values implemented in hardware registers in RTL Verilog;

FIG. 11A, comprising FIGS. 11A1-11A2, shows an initial C-language description having pointers and pointer indirection;

US 6,226,776 B1

9

FIG. 11B, comprising FIGS. 11B1–11B8, shows HDL created by compiling the C-type program shown in FIG. 11A, illustrating a non-addressable variable “p” implemented in a hardware register, addressable variables “pTree” and “Nodes” implemented in memory locations identified by addresses of “_pTree” and “_Nodes” in externally addressable hardware memory, and “while” loops, “if-then-else” statements, “return” statements, and function calls in RTL Verilog;

FIG. 12A shows an initial C-language description with a callable function;

FIG. 12B shows HDL created by compiling the C-type program shown in FIG. 12A, illustrating detection of the event “event_occurred” on the occurrence of the positive edge of a clock signal using “always” statements;

FIG. 13A shows an initial C-language description having a callable function and a pragma which directs compilation for asynchronous event detection;

FIG. 13B shows HDL created by compiling the C-type program shown in FIG. 13A illustrating detection of the event “event_occurred” when the signal “run_on_event” transitions from false to true, which occurs a synchronously with the positive edge of a clock signal;

FIG. 14A shows an initial C-language description with parallel processing of two complex functions (“func1” and “func2”) which do not interfere with each other;

FIG. 14B, comprising FIGS. 14B1–14B2, shows HDL created by compiling the C-type program shown in FIG. 14A, illustrating parallel processing of two complex non-interfering functions with two state machines implemented in RTL Verilog running simultaneously;

FIG. 15A shows an initial C-language description having variable argument function calls;

FIG. 15B, comprising FIGS. 15B1–15B7, shows HDL created by compiling the C-type program shown in FIG. 15A, illustrating variable argument function calls using a temporary parameter storage location in addressable memory implemented in RTL Verilog and using on-chip Xilinx RAM macros;

FIG. 16A shows an initial C-language description having a recursive function and recursive function call;

FIG. 16B, comprising FIGS. 16B1–16B5, shows HDL created by compiling the C-type program shown in FIG. 16A, illustrating a recursive function call to “MoveRing” utilizing addressable memory and a “stack_pointer” hardware register implemented in RTL Verilog;

FIG. 17A shows an initial C-language description having a function-call-by-pointer;

FIG. 17B, comprising FIGS. 17B1–17B2, shows HDL created by compiling the C-type program shown in FIG. 17A, illustrating the C-language pointer function implemented using a hardware register “state” assignment value corresponding to a first state of function “_main_f” in RTL Verilog;

FIG. 18A shows an initial C-language description having addressable variables;

FIG. 18B, comprising FIGS. 18B1–18B5, shows HDL created by compiling the C-type program shown in FIG. 18A, illustrating addressable memory implemented utilizing a hardware register file in RTL Verilog;

FIG. 19A shows an initial C-language description having multiple interdependent C-type functions;

FIG. 19B, comprising FIGS. 19B1–19B3, shows HDL created by compiling the C-type program shown in FIG.

10

19A, comprising multiple state machines implemented in RTL Verilog that operate in a data processing pipeline;

FIG. 19C is a block diagram of the data processing pipeline implemented by the RTL Verilog shown in FIG. 19B;

FIG. 20A shows an initial C-language description for a blinking light counter hardware design implemented utilizing a Xilinx FPGA on an APS-X84 hardware board;

FIG. 20B, comprising FIGS. 20B1–20B2, shows HDL created by compiling the C-type program shown in FIG. 20A for a blinking light counter hardware design implemented utilizing a Xilinx FPGA on an APS-X84 hardware board;

FIG. 21A shows an initial C-language description for a callable function;

FIG. 21B shows HDL created by compiling the C-type program shown in FIG. 21A illustrating the use of a state machine to implement a cumulative adder function in RTL Verilog;

FIG. 21C shows HDL created by compiling the C-type program shown in FIG. 21A, illustrating the use of a state machine to implement the cumulative adder function in VHDL;

FIG. 22A shows an initial C-language description for reading a memory word located at the memory address referenced by the variable parameter “p” which has an ANSI C “pointer” type;

FIG. 22B shows HDL created by compiling the C-type program shown in FIG. 22A, illustrating reading a memory word located at the memory address referenced by the variable parameter “p”, which has an ANSI C “pointer” type, in RTL Verilog;

FIG. 22C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers and control logic for reading 4-bit data located in external addressable memory at the location pointed by the 4-bit address, illustrating the compilation of the C-type program shown in FIG. 22A to a gate-level description;

FIG. 23A shows an initial C-language description for writing the content of the parameter variable “d” into a memory word located at the memory address referenced by the variable parameter “p” which has an ANSI C “pointer” type;

FIG. 23B shows HDL created by compiling the C-type program shown in FIG. 23A, which writes the content of the parameter variable “d” into a memory word located at the memory address referenced by the variable parameter “p”, which has an ANSI C “pointer” type, in RTL Verilog;

FIG. 23C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers and control logic for writing 4-bit data into the external addressable memory at the addressable memory location pointed by the 4-bit address, illustrating the compilation of the C-type program shown in FIG. 23A to a gate-level description;

FIG. 24A shows an initial C-language description for reading a structure member by using pointer to structure;

FIG. 24B shows HDL created by compiling the C-type program shown in FIG. 24A for reading a structure member by using pointer to structure in RTL Verilog;

FIG. 24C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers, control logic, and an adder for providing a constant offset to a given address for reading 4-bit data from the external addressable memory at the addressable memory location pointed by the 4-bit address plus offset, illustrating the compilation of the C-type program shown in FIG. 24A to a gate-level description;

US 6,226,776 B1

11

FIG. 25A shows an initial C-language description for writing into a structure member using pointer to structure;

FIG. 25B shows HDL created by compiling the C-type program shown in FIG. 25A for writing into a structure member using pointer to structure in RTL Verilog;

FIG. 25C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers, control logic, and an adder for providing a constant offset to a given address for writing 4-bit data into the external addressable memory at the addressable memory location pointed by the 4-bit address plus offset, illustrating the compilation of the C-type program shown in FIG. 25A to a gate-level description;

FIG. 26A shows an initial C-language description for reading a structure member from memory using a dot (".") operation;

FIG. 26B shows HDL created by compiling the C-type program shown in FIG. 26A for reading a structure member from memory using a dot (".") operation in RTL Verilog;

FIG. 26C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers and control logic for reading 4-bit data from the external addressable memory at an address fixed at compile time on output wires "A0-1", illustrating the compilation of the C-type program shown in FIG. 26A to a gate-level description;

FIG. 27A shows an initial C-language description for writing to a structure member located in memory;

FIG. 27B shows HDL created by compiling the C-type program shown in FIG. 27A for writing to a structure member located in memory in RTL Verilog; and

FIG. 27C shows a gate-level schematic comprised of clock enabled flip-flop hardware registers and control logic for writing 4-bit data into the external addressable memory at an address fixed at compile time on output wires "A0-1", illustrating the compilation of the C-type program shown in FIG. 27A to a gate-level description.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The computer aided hardware design tool in accordance with the various embodiments of the invention is executed on a computer 12, as shown in FIG. 1. The computer aided hardware design tool in accordance with one embodiment of the invention is preferably a 32-bit electronic computer aided design (ECAD) application compatible with a Microsoft Windows 95 or Microsoft NT 3.51 or later operating system available from Microsoft Corporation located in Redmond, Wash. The computer 12 comprises a minimum of 16 MB of random access memory (RAM) and preferably include 32 MB of RAM. The computer 12 also comprises a hard disk drive having 40 MB of free storage space available.

In an alternative embodiment, the computer aided hardware design tool can be available by connecting to the web site of the supplier of the computer aided hardware design tool. The computer 12 is provided with an Internet connection, such as a modem, for connection to the web site of the supplier of the computer aided hardware design tool. The computer aided hardware design tool can be ported to the web and executed on a web server. Therefore, the requirements for computer 12 would be reduced.

As shown in FIG. 1, means for displaying information preferably in the form of a monitor 14 connected to computer 12 is also provided. The monitor 14 can be a 640x480, 8-bit (256 colors) VGA monitor and is preferably an 800x600, 24-bit (16 million colors) SVGA monitor. The com-

12

puter 12 is also preferably connected to a CD-ROM drive 16. As shown in FIG. 1, a keyboard 18 and a mouse 20 are provided for entry of an algorithmic representation for a hardware design in a high-level programming language, such as ANSI C, as well as mouse-driven navigation between menus displayed by the computer aided hardware design tool. The mouse 20 also enables persons utilizing the computer aided hardware design tool (referred to hereafter as "users") to select simulation of a hardware design prior to implementation of a hardware design.

The computer aided hardware design tool in accordance with the invention enables a user to design hardware utilizing a high-level programming language, such as ANSI C. The fundamentals of ANSI C programming language are described, for example, in Kernighan, B. W., and Ritchie, D. M., *The C Programming Language*, Prentice Hall: New Jersey, 2d Ed., 1988. The computer aided hardware design tool of the invention then compiles the high-level program for the hardware design into a register transfer level (RTL) hardware description language (HDL) design which can be synthesized. The HDL may be, for example, Verilog, the fundamentals of which are described in Palnitkar, S., *Verilog HDL: A Guide to Digital Design and Synthesis*, SunSoft Press, Prentice Hall, 1996. For general information regarding compilers, see, for example, Aho, A. V., and Ullman, J. D., *Principles of Compiler Design*, Addison-Wesley Publishing Company, 2d Printing, March, 1978, and Hunter, R., *The Design and Construction of Compilers*, John Wiley & Sons, 1981.

The method in accordance with the invention for rendering a hardware design in a standard high-level programming language and converting that design into an actual hardware implementation will now be described. In accordance with one embodiment of the method of the invention, a user initially renders a preliminary hardware design as an algorithmic representation in a given standard high-level programming language. If the high-level programming language is not a C-type programming language, the high-level programming language preliminary hardware design is next translated from a preliminary hardware design rendered in the high-level programming language to a preliminary hardware design in a C-type programming language. Then, the C-type programming language preliminary hardware design is converted to a gate-level hardware representation which can be reduced to an actual hardware implementation, such as an FPGA or other programmable logic or an ASIC, using conventional physical design tools. This enables hardware designs rendered in virtually any standard high-level programming language to be implemented in hardware.

Considered in more detail, one embodiment of the method in accordance with the invention is shown in FIG. 2. A user can initially create an algorithmic representation corresponding to a preliminary hardware design using a given standard high-level programming language, as indicated by the step 30. Preferably, the high-level programming language is a programming language that can be translated into a C-type programming language, such as ANSI C, utilizing a conventional compilation program. Standard high-level programming languages that can be compiled to a C-type programming language utilizing conventional compilers include APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

US 6,226,776 B1

13

As indicated by the step 32 shown in FIG. 2, the high-level programming language algorithmic representation corresponding to the preliminary hardware design produced at step 30 is next translated into a C-type programming language preliminary hardware design. Translation results in a C-type programming language preliminary hardware design file, as indicated by the numeral 34 shown in FIG. 2. Preferably, the C-type programming language into which the high-level programming language preliminary hardware design is translated is ANSI C, such that an ANSI C preliminary hardware design file is produced at step 34.

Alternatively, the user can initially create the algorithmic representation corresponding to the preliminary hardware design in a C-type programming language, as indicated by the step 36 shown in FIG. 2. Preferably, the C-type programming language is ANSI C, such that an ANSI C preliminary hardware design file is produced at step 34. Therefore, the method in accordance with the invention enables a user to directly design hardware in a C-type programming language, preferably ANSI C programming language. This avoids the need at step 32 to translate from a different high-level programming language to a C-type programming language, for example, ANSI C.

As shown in FIG. 2, the C-type programming language preliminary hardware design file 34 can be run through a C compiler, as indicated by the step 38 shown in FIG. 2. This enables the user to verify or debug the C-type programming language preliminary hardware design.

Whether an algorithmic representation of a preliminary hardware design is initially created in a given standard high-level programming language translatable to a C-type programming language or created directly in the C-type programming language, such as ANSI C, the preliminary hardware design is compiled into a hardware description language (HDL) synthesizable design, as indicated by the step 40 shown in FIG. 2. For example, the C-type programming language preliminary hardware design can be compiled into a Verilog synthesizable design to produce an RTL hardware design file 42, as shown in FIG. 2. The compilation of the C-type programming language preliminary hardware design into a synthesizable HDL design will be described in more detail below.

After the HDL design is produced, the method in accordance with the invention preferably comprises the step of simulating the HDL synthesizable design prior to synthesizing the HDL design, as indicated by the step 44 shown in FIG. 2. If the user desires to modify the preliminary hardware design in view of the simulation, the user renders a modified hardware design using either the given standard high-level programming language at step 30 or the C-type programming language, such as ANSI C at step 36.

After an acceptable HDL design is obtained, the RTL hardware design file 42 representative of the HDL design is synthesized into a gate-level hardware representation using a conventional synthesis program to interpret the HDL design, as indicated by the step 46 shown in FIG. 2. This typically requires preselection of a particular hardware technology, such as an FPGA or ASIC, prior to the execution of the synthesis program. Synthesis programs are commercially available and do not constitute the invention per se. Therefore, the synthesis programs will not be described in detail.

In order to produce an actual hardware implementation, the method in accordance with the invention preferably further comprises the step of using conventional physical design tools to implement the gate-level hardware represen-

14

tation as an actual hardware implementation. These physical design tools include a place and route program. In the case of an FPGA, the physical design tools also include a hardware configuration system which processes the file produced by the place and route program and creates a bit stream to personalize the FPGA, as indicated by the step 48 shown in FIG. 2. In the case of an ASIC, the physical design tools include a place and route program for creating a file for patterning an integrated circuit mask for use in fabricating the ASIC, as indicated by the step 50 shown in FIG. 2. Physical design tools utilized to produce actual hardware implementations are commercially available and do not constitute the invention per se. Therefore, the physical design tools will not be described in detail. As shown in FIG. 2, an actual hardware implementation is produced, as indicated by the numeral 52.

A detailed description of the steps required for compiling a C-type programming language algorithmic representation for a preliminary hardware design into an HDL synthesizable design will now be provided by way of example. In the example, the C-type programming language is ANSI C and the HDL is preferably Verilog. However, it will be understood by persons skilled in the art that the principles of compiling a C-type programming language algorithmic representation for a preliminary hardware design into an HDL synthesizable design apply to other C-type programming languages besides ANSI C. For example, the C-type programming language may alternatively be any standard C-type programming language including ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C. Moreover, the HDL synthesizable design into which the C-type programming language algorithmic representation for the preliminary hardware design is translated may be other than Verilog. For example, the HDL may alternatively be any standard HDL including VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

The following describes compilation of the operations and functions that comprise ANSI C high-level programming language needed for hardware design into an exemplary HDL, namely, Verilog. As will be described later, the HDL may also be a different HDL, such as VHDL.

The method in accordance with the invention comprises mapping predetermined C-type programming language expressions to functionally equivalent HDL program language expressions. FIG. 3A illustrates a simple expression function in ANSI C programming language, and FIG. 3B represents the function shown in FIG. 3A compiled into RTL Verilog HDL.

In this regard, statements in ANSI C high-level programming language used to render hardware designs vary. For example, various ANSI C functions can be classified "simple expression functions," such as "func1" and "func2" shown in FIG. 3A. These functions have no side effects, do not require static memory access in HDL, and do not require a state machine in HDL to run. The determination of whether or not "simple expression functions" are present is performed by the code that appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 5052-5232. Consequently, these functions can be compiled into simple RTL Verilog continuous assignments "result_func1" and "result_func2", as shown in FIG. 3B. Compilation of the ANSI C expressions to functionally equivalent HDL expressions is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5889-5990.

Furthermore, FIG. 3A illustrates C-type programming language expressions "a+b^c<<3" and "(a^b) & c". In these

US 6,226,776 B1

15

expressions, "a", "b", and "c" are integer variables. FIG. 3B illustrates that these expressions are mapped into RTL Verilog expressions "`((param_func1_a+param_func1_b)-(param_func1_c <<16'd3))`" and "`((param_func2_a^param_func2_b) & param_func2_c)`", respectively. Also, integer variables "a", "b", and "c" in FIG. 3A are mapped onto wires "param_func1_a", "param_func1_b", and "param_func1_c" for ANSI C function "func1" and wires "param_func2_a", "param_func2_b", and "param_func2_c" for ANSI C function "func2", respectively. This is because the ANSI C functions "func1" and "func2" are simple C-type functions which do not require implementation in RTL Verilog as state machines. The same mapping would occur if variables "a", "b", and "c" were defined as ANSI C floating point, pointer, enum, or union variables.

Additionally, FIGS. 3A and 3B illustrate assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design. The determination of whether or not C-type variables are to be mapped onto input/output wires of the generated RTL Verilog module is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4802–5001. For example, parameters "a", "b", and "c" of the function "func1" shown in FIG. 3A are assigned to input wires, namely, "param_func1_a", "param_func1_b", and "param_func1_c", respectively, of the module generated in RTL Verilog, as shown in FIG. 3B. Also, the result of the function "func1" (FIG. 3A) is assigned to an output wire "result_func1" in RTL Verilog (FIG. 3B).

FIGS. 3A and 3B also illustrate a plurality of selected C-type functions that can execute simultaneously, which are compiled into a plurality of executable HDL program language expressions that operate in parallel. The determination of whether or not a plurality of simultaneously executable C-type functions is present is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 5052–5232. Compilation of the plurality of simultaneously executable C-type functions to a plurality of HDL expressions that operate in parallel is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5889–5990.

FIGS. 4A and 4B also illustrate mapping predetermined C-type programming language expressions to functionally equivalent HDL program language expressions. FIGS. 4A and 4B show compilation from ANSI C (FIG. 4A) into RTL Verilog HDL (FIG. 4B) where no HDL state machine is required to implement the conversion of ANSI C functions. These ANSI C functions have no side effects, do not require static memory access in HDL, and do not require a state machine in HDL to run. The advantage of utilizing state machines only when they are needed is that the compiled HDL design runs faster in hardware.

However, in contrast to the ANSI C functions shown in FIG. 3A, the ANSI C functions shown in FIG. 4A cannot be classified as "simple expression functions," because they have data conversions or comprise at least one "if-then-else" statement. For example, "func1" shown in FIG. 4A contains a comparison operator that has different semantics in ANSI C than in RTL Verilog (i.e., utilizes temporary registers in Verilog). Therefore, "func1" cannot be classified as a "simple expression function," but it can be classified as a "function outside the HDL state machine." These functions, such as "func1", cannot be converted into one continuous assignment in HDL but must instead be converted into multiple HDL statements within an "always" block. Note

16

that FIGS. 4A and 4B also illustrate assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design, as described earlier in connection with FIGS. 3A and 3B.

FIGS. 5A and 5B illustrate compiling a C-type program control flow into an HDL state machine. FIG. 5A shows a simple "for" loop in ANSI C, and FIG. 5B shows compilation of the "for" loop into RTL Verilog HDL. The determination of whether or not a C-type program control flow that is to be converted to an HDL state machine is present is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 3735–4271.

As shown in FIG. 5B, the "for" loop is implemented in RTL Verilog using a state machine. A state machine is utilized to implement the ANSI C program control flow, because the state machine provides the mechanism to execute the ANSI C operators in the required sequence. Compilation of the C-type program control flow into the HDL state machine is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5930–6250.

FIGS. 5A and 5B also illustrate mapping non-addressable variables onto hardware registers formed in HDL as a result of compilation. The compilation is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4802–5001. The non-addressable variables "n", "i", and "sum" employed in the ANSI C program shown in FIG. 5A are mapped onto hardware registers "_sum1_n", "_2_sum_i", and "_2_sum1_sum", respectively, shown in FIG. 5B.

Additionally, FIGS. 5A and 5B illustrate mapping addressable variables onto addressable on-chip static memory formed in HDL as a result of compilation. The compilation is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4802–5001. FIG. 5A shows that the ANSI C program includes an array "array" which is mapped onto addressable on-chip static memory having a starting memory address "`_5_main_array`", as shown in FIG. 5B.

FIGS. 5A and 5B also illustrate assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design. The determination of whether or not C-type variables are to be mapped onto input/output wires of the generated RTL Verilog module is performed by the code which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4802–5001. For example, parameter "n" of the function "sum1" shown in FIG. 5A is assigned to an input wire, namely, "param_sum1_n", of the module generated in RTL Verilog, as shown in FIG. 5B. Also, the result of the function "sum1" (FIG. 5A) is assigned to an output wire "result_sum1" in RTL Verilog (FIG. 5B).

FIGS. 6A and 6B illustrate compilation of standard ANSI C input/output (I/O) function calls into RTL Verilog HDL. The I/O function arguments (namely, "scanf" and "printf" arguments) defined in the ANSI C program shown in FIG. 6A are assigned to specific wires in RTL Verilog, as shown in FIG. 6B.

The input function call "scanf" for reading data into ANSI C variables "a" and "b" shown in FIG. 6A is compiled into RTL Verilog for reading data from input wires, namely, "scanf_0_1_line_19_a" and "scanf_0_2_line_19_b", into registers "`_3_main_a`" and "`_3_main_b`", respectively, as shown FIG. 6B. Accordingly, variables "a" and "b" in the ANSI C program (FIG. 6A) are mapped into registers "`_3_main_a`" and "`_3_main_b`" in RTL Verilog

US 6,226,776 B1

17

(FIG. 6B). The output function call “printf” for writing the result of the ANSI C expression “ $a+b$ ” shown in FIG. 6A is compiled into writing the result of the HDL expression “ $_3_main_a+_3_main_b$ ” to an output wire, namely, “printf_1_1_line_20” in RTL Verilog, as shown FIG. 6B.

As also shown in FIG. 6B, a Verilog statement “\$write (“ $a+b=% d\n$ ”, printf_1_1_line_20);” is preferably generated for the purpose of debugging the generated Verilog code. This statement is not used in connection with synthesis of the HDL design.

FIGS. 7A and 7B illustrate compiling multiple occurrences of a complicated C-type mathematical operation into a given HDL functional block that implements such an operation. The functional block is invoked on each occurrence by different states driven by the state machine.

FIG. 7A shows a C-type programming language expression “ $a/b+c/d$ ”. This expression includes two divisions, namely, “ a/b ” and “ c/d ”. These divisions share the same divider unit implemented in RTL Verilog HDL. The divider unit is implemented in RTL Verilog, as follows. First, a divider module is defined, namely, “sdivmod16”. Second, the divider module “sdivmod16” is instantiated in the module “DivEx” comprising the state machine implemented in RTL Verilog. Third, the shared divider unit is driven employing the different states of the state machine, as shown in FIG. 7B. Since a single divider module is preferably utilized to reduce required hardware resources, the state machine is needed to sequentially multiplex arithmetic parameters to the inputs of the shared divider module in synchronization with the hardware clock. Compilation of the complicated C-type mathematical operation for division into a given HDL functional block for division that is invoked on the occurrence of different states of the state machine is performed by the code shown in the source code listing which appears in microfiche Appendix A in files GDIVMOD.CPP at lines 8487–9243, GEXPR.CPP at lines 7324–7646, and GFLOAT.CPP at lines 9244–9738. Based on the compilation of a division operation disclosed in FIGS. 7A and 7B, it should be apparent to persons skilled in the art how to handle other complicated arithmetic and logic operations, such as signed and unsigned integer multiplication, signed and unsigned integer remainder, variable-shift left shift, variable-shift right shift with or without sign extension, etc.

ANSI C variables “ a ”, “ b ”, “ c ”, and “ d ” shown in FIG. 7A are defined as integer variables. These variables are mapped onto wires “param_func_a”, “param_func_b”, “param_func_c”, and “param_func_d” of the RTL Verilog module, respectively. The same mapping would occur if variables “ a ”, “ b ”, “ c ”, and “ d ” were defined as ANSI C floating point, pointer, enum, or union variables.

ANSI C does not support the concept of “pipelining.” However, it has been determined that ANSI C constructs can be utilized to define a hardware data processing pipeline.

As shown in FIGS. 8A and 8B, the method in accordance with the invention may also comprise compiling a plurality of interdependent C-type functions into a plurality of executable HDL program language expressions that operate either simultaneously or sequentially in a data processing pipeline. C-type functions are interdependent when the input of one of the functions is dependent upon the output of another function. FIGS. 8A and 8B illustrate compilation of three simple ANSI C functions (FIG. 8A) into a data processing pipeline implemented in RTL Verilog HDL (FIG. 8B).

As shown in FIG. 8C, the data processing pipeline consists of three stages. The first stage of the pipeline corresponds to the ANSI C function “pipeline_stage_1”. This

18

first stage is an adder which adds the value “1” to the value from the input wire “ $_data_in$ ”, which corresponds to the ANSI C variable “ $data_in$ ”, and writes the sum into the hardware register “ $_out1$ ”, which corresponds to the ANSI C variable “ $out1$ ”. The second stage of the pipeline corresponds to the ANSI C function “ $pipeline_stage_2$ ” which uses the output “ $_out1$ ” and adds that value to itself and writes the sum into the hardware register “ $_out2$ ” which corresponds to the ANSI C variable “ $out2$ ”. The third stage of the pipeline corresponds to the ANSI C function of “ $pipeline_stage_3$ ” which uses the output “ $_out2$ ” of the second stage and performs an “XOR” logic function with the value “1234” and writes the result onto the output wire “ $_data_out$ ” which corresponds to the ANSI C variable “ $data_out$ ”. FIG. 8B illustrates that the three stages of the data processing pipeline are implemented by three “always” blocks in RTL Verilog, which operate simultaneously. Alternatively, the stages of the data processing pipeline implemented in RTL Verilog may be operated sequentially depending upon usage of input signals “run_pipeline_1”, “run_pipeline_2”, and “run_pipeline_3”. The compilation is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4896–4950.

Variable “ $data_in$ ” shown in the ANSI C program (FIG. 8A) has “external scope,” as defined in the ANSI C standard. The address of “ $data_in$ ” is not used by any C-type function or for C-type initialization of any variable. Furthermore, “ $data_in$ ” is never used as an 1-value. That is, “ $data_in$ ” is used in the ANSI C program as a readable variable, but it is not used as a writable variable. Also, “ $data_in$ ” is not initialized in the ANSI C program.

Because “ $data_in$ ” has these attributes, it is mapped onto an input wire “ $_data_in$ ” of an HDL module, as shown in FIG. 8B. The input wire is utilized as an input wire of a hardware pipeline stage implemented in RTL Verilog (FIG. 8B).

Variable “ $data_out$ ” shown in the ANSI C program (FIG. 8A) also has “external scope.” The address of “ $data_out$ ” is not used by any C-type function or for C-type initialization of any variable. Furthermore, “ $data_out$ ” is used only as an 1-value. That is, “ $data_out$ ” is used in the ANSI C program as a writable variable, but it is not used as a readable variable. Also, “ $data_out$ ” is not initialized in the ANSI C program.

Since “ $data_out$ ” has the above attributes, it is mapped onto an output wire “ $_data_out$ ” of the HDL module, as shown in FIG. 8B. The output wire is utilized as an output wire of the hardware pipeline stage implemented in RTL Verilog (FIG. 8B).

FIGS. 8A and 8B illustrate a simple pipeline. That is, the ANSI C program shown in FIG. 8A is compiled into RTL Verilog without the need for an HDL state machine.

FIGS. 9A and 9B illustrate floating point arithmetic and double precision floating point arithmetic in ANSI C (FIG. 9A) and compilation into RTL Verilog HDL (FIG. 9B). This demonstrates that the method in accordance with the invention can convert floating point operations into a state-machine-based protocol of transactions to an external floating point arithmetic unit (FPU) preferably implemented in adjunct hardware. Since FPUs are commercially available, definition of an FPU in the RTL Verilog generated during compilation of the ANSI C arithmetic operations is not required. Instead, all that is needed is the assignment of output and input wires to such an FPU, which defines an interface between the RTL Verilog generated design and the FPU.

US 6,226,776 B1

19

As shown in FIGS. 9A and 9B, each of the ANSI C arithmetic operations is compiled into a sequence of states of the state machine in RTL Verilog. During a first state, the parameters needed for the floating point operations are written to output wires assigned in RTL Verilog, which are read by the external floating point arithmetic unit. During a second state, the state machine awaits completion of performance of the arithmetic operations by the arithmetic unit. During a third state, the results of the floating point arithmetic operations are written by the arithmetic unit onto the input wires defined in RTL Verilog and read.

FIGS. 9A and 9B show compilation from ANSI C into RTL Verilog for subtraction from zero (unary-), addition, multiplication, subtraction, division, and conversion operations, which involve floating point arithmetic. Therefore, in view of the methodology shown in FIGS. 7A, 7B, 9A, and 9B, it will be apparent to persons skilled in the art that all complicated integer and floating point operations can be compiled into RTL Verilog.

FIGS. 10A and 10B illustrate compilation of descriptions with structures, including structure assignment, structure function parameters, and structure function return values, which are commonly employed to implement algorithms, in ANSI C (FIG. 10A) and following compilation into RTL Verilog HDL (FIG. 10B). FIG. 10A shows the addition of two complex numbers implemented as ANSI C structures. The expression “z=add (x, add (x,y))” contains the structure assignment to structure “z”, and structures “x” and “y” are employed as structure function parameters of the function “add”. Function “add” returns a structure return value which is equal to structure “c”. FIG. 10A also illustrates structure members “real” and “image”, which are employed both as l-value (value on the left-hand side of the assignment operation) and r-value (value on the right-hand side of the assignment operation). FIG. 10B illustrates that structure function parameters and structure function return values are implemented utilizing wide (i.e., 32-bit) hardware registers which can store the entire structure, including structure members (i.e., “real” and “image”). Temporary wide (i.e., 32-bit) hardware registers are also utilized for the structure assignment translation to read the r-value structure from external memory and to write the l-value structure to the external memory. Compilation of structure assignment and structure return values and assignment of structure function parameters is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GEXPR.CPP at lines 8101–8255. It will be understood by persons skilled in the art that the same compilation process can be applied to C-type programming language structure assignment, structure function parameters, and structure function return values in C-type programs which do not require implementation of a state machine in RTL Verilog.

FIGS. 11A and 11B illustrate compiling C-type pointers and pointer indirection into an HDL state-machine-based memory access protocol. FIG. 11A shows C-type pointers, such as local variable “p”, return value of function “FindNode”, global variable “pTree”, and structure members “pLeft” and “pRight”. On the one hand, local variable “p” is not addressable, that is, its address is never read. Therefore, this non-addressable variable is mapped onto a hardware register formed in RTL Verilog HDL, namely, “_7_FindNode_p”, as shown in FIG. 11B. On the other hand, variable “pTree” is addressable, that is, its address is read in the expression “p=& pTree;”, as shown in FIG. 11A. Therefore, this addressable variable is mapped onto a memory location identified by the address “_pTree” in external addressable hardware memory, as shown in FIG.

20

11B. Compilation of C-type pointers and pointer indirection into an HDL state-machine-based memory access protocol is performed by the code shown in the source code listing which appears in microfiche Appendix A in files GRW.CPP at lines 10040–10438 and GEXPR.CPP at lines 6724–6750, 7252–7282, and 8101–8255.

Also, “Nodes” is an addressable variable because it is a C-type programming language array. Therefore, this addressable variable is mapped onto a memory location identified by the address “_Nodes” in external addressable hardware memory, as shown in FIG. 11B.

FIG. 11A also illustrates a “while” loop, “if-then-else” statement, “return”, and function call (i.e., “FindNode” is called by the function “FindValue”) in ANSI C programming language, and FIG. 11B illustrates implementation of these as a state machine in RTL Verilog. FIGS. 11A and 11B also provide another illustration of the handling of C-type programming language structures, including translation of the operation “->” (i.e., structure member dereferencing by pointer).

The method of the invention may also comprise compiling a callable C-type function into a functionally equivalent HDL program language expression that is executed synchronously on the occurrence of an external event. FIGS. 12A and 12B illustrate a simple case of synchronous event detection in the case where the HDL expression does not require a state machine. The C-type function “on_event” represents any executable C-type function, which is called for execution by another C-type function, for example, “main”. This call is converted into RTL Verilog HDL that is executable on the occurrence of an external event, such as assertion of a signal on an input wire. The determination of whether or not a callable C-type function is present is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 5052–5232. Thus, the C-type function “on_event” (FIG. 12A) is compiled into RTL Verilog that is executed when signal “run_on_event” is asserted true (FIG. 12B). Compilation of the callable C-type function into a functionally equivalent HDL program language expression which is executed synchronously on the occurrence of an external event is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5889–5990. Event detection in this implementation is synchronous, that is, the event is detected only on a positive edge of a clock signal. It will be understood by persons skilled in the art that the same compilation process can be applied to a callable C-type function which requires implementation of a state machine in RTL Verilog in the case in which the state machine is executed synchronously on the occurrence of an external event. The following example will illustrate asynchronous event detection.

FIGS. 13A and 13B illustrate compiling a callable C-type function into a functionally equivalent HDL program language expression that is executed a synchronously after the occurrence of an external event. FIGS. 13A and 13B illustrate a simple case of asynchronous event detection in the case where the HDL expression does not require a state machine. The determination of whether or not a callable C-type function is present is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 5052–5232.

As in the case of FIG. 12A, the C-type function “on_event” shown in FIG. 13A again represents any executable C-type function which is called for execution by another

US 6,226,776 B1

21

C-type function, for example, "main." However, as shown in FIG. 13B, this call is converted into RTL Verilog HDL that is executable following occurrence of an external event, such as transition of a signal on an input wire. That is, the pragma causes the C-type function "on_event" (FIG. 13A) to be compiled into RTL Verilog which is executed when signal "run_on_event" transitions from false to true (FIG. 13B). This occurs synchronously relative to the occurrence of the positive edge of a clock signal ("posedge clock"). Compilation of the callable C-type function into functionally equivalent HDL program language expression which is executed synchronously on the occurrence of an external event is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5889-5990. Note that asynchronous event processing must handle timing constraints that do not occur in the synchronous event implementation shown in FIGS. 12A and 12B.

It will be understood by persons skilled in the art that a similar compilation process can be applied to a callable C-type function that requires implementation of a state machine in RTL Verilog HDL. In the case in which the state machine is executed synchronously with respect to the occurrence of an external event, instantiation of a flip-flop is required in RTL Verilog to store the occurrence of the event. If this flip-flop is set, the state machine runs on the next-occurring positive edge of a clock signal.

FIGS. 14A and 14B illustrate parallel processing of two complex C-type functions and two state machines running simultaneously. These functions ("func1" and "func2") do not interfere with each other, which means they: 1) do not call each other; 2) do not share external or on-chip static memory; 3) do not share an HDL functional block that performs a complicated mathematical operation; and 4) do not write into the same hardware register. Because of this fact, these functions can be performed simultaneously. External logic can safely assert signals "run_func1" and "run_func2" at any time independently of each other, as shown in FIG. 14B. Compilation of the complex C-type functions that can execute simultaneously into plurality of HDL state machines that operate in parallel is performed by the code shown in the source code listing which appears in microfiche Appendix A in file GCODE.CPP at lines 5615-6251. Since these functions require state machines to implement them, their execution is synchronous with the positive edge of a clock signal.

FIGS. 15A and 15B illustrate compilation of ANSI C variable argument functions and variable argument function calls into RTL Verilog HDL. As shown in FIG. 15A, the ANSI C variable argument function "average" is called from another ANSI C function "main". In order to convert the variable argument function call "average" to RTL Verilog, a temporary parameter storage location in addressable memory is utilized. This temporary storage location resides immediately after the location of the last fixed parameter, namely, "'_average_first'", of the variable argument function "average", as shown in FIG. 15B. When function "main" calls function "average", function "main" sets the parameters for function "average" by writing values for those parameters into the addressable memory at the temporary parameter storage location.

The method in accordance with the invention also supports the use of ANSI C recursive function calls. The C-type program is parsed to determine whether or not a direct or indirect recursive function call is present. That is, a global analysis of the C-type program is performed to determine which functions are recursive. Any identifiable direct or

22

indirect recursive function call is compiled into an HDL state machine which has an interface to a stack implemented utilizing either external or internal memory or, alternatively, an array of hardware registers. The generated RTL Verilog module has a stack pointer implemented utilizing a hardware register employed to store and restore local variables of the recursive function and other recursive function information.

FIGS. 16A and 16B illustrate compilation of an ANSI C recursive function and recursive function call into RTL Verilog HDL. As shown in FIG. 16A, the ANSI C recursive function "MoveRing" is called from itself and from another ANSI C function "Hanoi". In order to translate the recursive function call to "MoveRing" into RTL Verilog:

1. A designated region of addressable memory at the location "recursion_stack" is established in RTL Verilog, as shown in FIG. 16B.

2. A designated hardware register, namely "stack_pointer" is defined in RTL Verilog, as also shown in FIG. 16B.

3. Before the recursive function "MoveRing" calls itself (i.e., the recursive function "MoveRing"), recursive function "MoveRing" saves its local context into the established stack which consists of the designated region of the addressable memory at the location "recursion_stack" and the designated hardware register, namely, "stack_pointer". The recursive function "MoveRing" local context contains recursive function "MoveRing" local variable and parameter registers, namely, "_MoveRing_nRings", "_MoveRing_Tower1", "_MoveRing_Tower2", "_MoveRing_Tower3", and the recursive function "MoveRing" return state register, namely, "return_state_MoveRing".

4. After the recursive function "MoveRing" call, the recursive function "MoveRing" restores its local context, as shown in FIG. 16B.

By storing and restoring its local context before and after the recursive call, respectively, the recursive function "MoveRing" avoids losing its local context. Before the recursive function calls itself, it saves the local context on the memory stack (an addressable memory). After executing the function, the recursive function restores its local context.

FIGS. 17A and 17B illustrate compilation of an ANSI C function-call-by-pointer into RTL Verilog HDL. As shown in FIG. 17A, the ANSI C functions "f1" and "f2" are called from the ANSI C function "main" using a pointer variable "f". Depending upon the value of parameter "n" of the ANSI C function "main", either pointer-to-function "f1" or pointer-to-function "f2" is assigned to the pointer variable "f". Then, the ANSI C function "main" calls one of "f1" or "f2" using the pointer variable "f". This ANSI C function-call-by-pointer is translated into RTL Verilog by assigning the hardware register "state" a value corresponding to the first state of the function, namely, "_main_f", as shown in FIG. 17B. The assigned value becomes the current state of the generated state machine. Depending upon the value of the "_main_n" hardware register in RTL Verilog, the pointer-to-function hardware register "_main_f" is assigned to either state "_f1" or to state "_f2" in RTL Verilog, which correspond to ANSI C functions "f1" and "f2", respectively.

FIGS. 18A and 18B illustrate mapping addressable variables defined in C-type programming language onto an addressable array of hardware registers formed in HDL. Specifically, "array" is an addressable variable, because it is a C-type programming language array, as shown in FIG. 18A. Therefore, this ANSI C array is mapped onto addressable memory implemented as an addressable array of

US 6,226,776 B1

23

hardware registers, namely, "memory" in RTL Verilog HDL, as shown in FIG. 18B. The compilation is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4802–5001. The starting address of the ANSI C "array" in the "memory" array in RTL Verilog shown in FIG. 18B is "offset_main_3_main_array".

The method in accordance with the invention may also comprise compiling a plurality of interdependent C-type functions into a plurality of HDL state machines that operate in a data processing pipeline. FIGS. 19A and 19B illustrate compilation of three ANSI C functions (FIG. 19A) into a data processing pipeline implemented in RTL Verilog HDL (FIG. 19B). In contrast to the three-stage data processing pipeline shown in FIG. 8C, the three-stage data processing pipeline shown in FIG. 19C comprises a separate state machine to implement each of the three stages of the data processing pipeline in RTL Verilog. Otherwise, the configuration of the data processing pipeline shown in FIG. 19C is similar to the configuration of the data processing pipeline shown in FIG. 8C. Although the implementation of the data processing pipeline in RTL Verilog shown in FIG. 19B includes a state machine for each stage, other ANSI C functions do not require the implementation of state machines, as in the case of the data processing pipeline shown in FIG. 8B. Alternatively, the data processing pipeline may comprise a combination of simple HDL functional blocks and state machines. The compilation is performed by the code shown in the source code listing which appears in microfiche Appendix A in file OSYMBOLS.CPP at lines 4896–4950.

FIGS. 20A and 20B illustrate that a hardware design originally rendered in ANSI C programming language may be implemented in actual hardware. The target hardware was a xilinx FPGA on an APS-X84 hardware board. The board was installed in a slot in a computer. The result was to implement a counter whose operation is evidenced by a blinking LCD on the board. The frequency of blinking is determined by the on-board clock rate and the user-specified terminal count. FIG. 20A illustrates the hardware design rendered in ANSI C programming language, and FIG. 20B shows the Verilog code after compilation.

FIG. 20B also shows configuration of an interface in the HDL synthesizable design for the gate-level hardware representation. That is, an additional interface Verilog module with wire assignments is needed to map the hardware I/O pins of the target hardware (APS-X84 FPGA) to the computer bus. This is shown in FIG. 20B after the comment "//APSX84.v—Verilog wrapper module which instantiates module Light.v.".

The HDL design may be simulated prior to and/or after the wire assignments are mapped to the I/O pins. For example, one of various simulation programs commercially available from Wellspring Solutions, Cadence Design Systems, Mentor Graphics Corporation, Synopsis, Viewlogic Systems, or Veri Best, Inc. may be used for simulation of the HDL design.

The hardware implementation may be performed using physical design tools to implement the gate-level representation produced by synthesizing the Verilog code using one of various synthesis programs commercially available from Synplicity, Synopsys, or Exemplar Logic. In this example, the physical design tool that was used is commercially available from Xilinx.

The foregoing describes compilation of a hardware design rendered in ANSI C high-level programming language into an HDL synthesizable design in RTL Verilog HDL. The

24

following example illustrates the compilation of a hardware design rendered in ANSI C into an HDL synthesizable design in both Verilog and VHDL. Therefore, the computer aided hardware design tool in accordance with the invention can provide compilation of a hardware design rendered in ANSI C high-level programming language into any of various HDLs.

FIGS. 21A–21C illustrate compilation of a hardware design from a high-level programming language to several HDLs commonly employed for hardware design. FIG. 21A shows a hardware design for a callable function rendered in ANSI C high-level programming language. FIG. 21A comprises the ANSI C text for the hardware design. FIG. 21B shows the same hardware design compiled into RTL Verilog HDL. FIG. 21B comprises the Verilog text for the hardware design. FIG. 21C shows the same hardware design compiled into another commercially available HDL, namely, VHDL. FIG. 21C comprises the VHDL text for the hardware design.

In accordance with one aspect of the invention, ANSI C pointer and structure operations are translated into transactions to external or on-chip static memory in a gate-level hardware representation. The known prior art does not process ANSI C pointers or structures and therefore disallows the use of pointers and structures in rendering a hardware design.

In accordance with a preferred embodiment of the invention, the ANSI C pointer operations are compiled into HDL that is synthesized to a gate-level hardware representation. FIG. 22A illustrates an ANSI C programming language description for reading a memory word located at the memory address referenced by the variable parameter "p" which has an ANSI C "pointer" type. FIG. 22B illustrates compilation of the ANSI C pointer operation "*" into RTL Verilog HDL that can be synthesized into a gate-level hardware design file, or netlist. FIG. 22C is a gate-level schematic diagram created from the netlist. FIG. 22C shows clock-enabled flip-flop hardware registers and control logic for reading 4-bit data located in external addressable memory at the location pointed by a 4-bit address on wires "PARAM_READ_P0-3", which illustrates the compilation of the C-type program to a gate-level hardware description. The external or on-chip memory is addressed by the address stored on wires "A0-3" at the Q outputs of the flip-flops shown in the lower portion of FIG. 22C. The memory returns the addressed data on wires "D0-3". The addressed data are clocked to the Q outputs of the flip-flops shown in the top portion of FIG. 22C onto wires "RESULT_READ0-3". Although FIGS. 22A–22C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

FIG. 23A illustrates an ANSI C programming language description for writing the content of the parameter variable "d" into a memory word located at the memory address referenced by the variable parameter "p" which has an ANSI C "pointer" type. FIG. 23B illustrates compilation of the ANSI C 4-bit-wide variable "d" into RTL Verilog HDL that can be synthesized into a netlist. FIG. 23C is a gate-level schematic diagram created from the netlist. FIG. 23C shows clock-enabled flip-flop hardware registers and control logic for writing 4-bit data to external addressable memory at the location pointed by a 4-bit address on wires "PARAM_WRITE_P0-3". The external memory is addressed by the values stored on wires "A0-3" at the Q outputs of the flip-flops shown in the top portion of FIG. 23C. The data to be stored are clocked from the wires "PARAM_WRITE_D0-3" to the D inputs of the flip-flops shown in the lower portion of FIG. 23C onto wires "D0-3". Although FIGS.

US 6,226,776 B1

25

23A–23C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

Also in accordance with a preferred embodiment of the invention, ANSI C structure operations are compiled into HDL that is synthesized to a gate-level hardware representation. FIG. 24A illustrates an ANSI C programming language description for reading a memory word located at the memory address referenced by the variable parameter “p” which has an ANSI C “pointer-to-structure” type and a fixed offset corresponding to the structure member “b”. FIG. 24B illustrates compilation of the ANSI C structure operation “->” into RTL Verilog HDL that can be synthesized into a gate-level hardware design file, or netlist. FIG. 24C is a gate-level schematic diagram created from the netlist. FIG. 24C shows clock-enabled flip-flop hardware registers, control logic, and an adder for providing a constant offset to a given address for reading 4-bit data located in external addressable memory at the location pointed by a 4-bit address on wires “PARAM_READ_MEMBER_B_P0-3” plus the offset provided by the adder, which illustrates the compilation of the C-type program to a gate-level hardware description. The external or on-chip memory is addressed by the address stored on wires “A0-3” at the Q outputs of the flip-flops shown in the upper portion of FIG. 24C. The memory returns the addressed data on wires “D0-3”. The addressed data are clocked to the Q outputs of the flip-flops shown in the lower portion of FIG. 24C onto wires “RESULT_READ_MEMBER_B0-3”. Although FIGS. 24A–24C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

FIG. 25A illustrates an ANSI C programming language description for writing the content of the parameter variable “n” into a memory word located at the memory address referenced by the variable parameter “p” which has an ANSI C “pointer-to-structure” type and a fixed offset corresponding to the structure member “b”. FIG. 25B illustrates compilation of the ANSI C structure operation “->” into RTL Verilog HDL that can be synthesized into a netlist. FIG. 25C is a gate-level schematic diagram created from the netlist. FIG. 25C shows clock-enabled flip-flop hardware registers, control logic, and an adder for providing a constant offset to a given address for writing 4-bit data to external addressable memory at the location pointed by a 4-bit address on wires “PARAM_WRITE_MEMBER_B_P0-3” plus the offset provided by the adder. The external memory is addressed by the values stored on wires “A0-3” at the Q outputs of the flip-flops shown in the top portion of FIG. 25C. The data to be stored are clocked from the wires “PARAM_WRITE_MEMBER_B_N0-3” to the D inputs of the flip-flops shown in the lower portion of FIG. 25C onto wires “D0-3”. Although FIGS. 25A–25C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

FIG. 26A illustrates an ANSI C programming language description for reading a memory word located at the memory address referenced by the fixed address determined at compile time of the “b” member of structure “s”. FIG. 26B illustrates compilation of the ANSI C structure operation “.” into RTL Verilog HDL that can be synthesized into a gate-level hardware design file, or netlist. FIG. 26C is a gate-level schematic diagram created from the netlist. FIG. 26C shows clock-enabled flip-flop hardware registers and control logic for reading 4-bit data located in external addressable memory at the location pointed by a fixed address “2”, which illustrates the compilation of the C-type program to a gate-level hardware description. The external or on-chip memory is addressed by the address stored on

26

wires “A0-1” at the Q outputs of the flip-flops shown in the middle portion of FIG. 26C. The memory returns the addressed data on wires “D0-3”. The addressed data are clocked to the Q outputs of the flip-flops shown in the lower portion of FIG. 26C onto wires “RESULT_READ_MEMBER_B0-3”. Although FIGS. 26A–26C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

FIG. 27A illustrates an ANSI C programming language description for writing the content of the parameter variable “n” into a memory word located at the memory address referenced by the fixed address determined at compile time of the “b” member of structure “s”. FIG. 27B illustrates compilation of the ANSI C structure operation “.” into RTL Verilog HDL that can be synthesized into a netlist. FIG. 27C is a gate-level schematic diagram created from the netlist. FIG. 27C shows clock-enabled flip-flop hardware registers and control logic for writing 4-bit data to external addressable memory at the location pointed by a fixed address “2”. The external memory is addressed by the values stored on wires “A0-1” at the Q outputs of the flip-flops shown in the middle portion of FIG. 27C. The data to be stored are clocked from the wires “PARAM_WRITE_MEMBER_B_N0-3” to the D inputs of the flip-flops shown in the lower portion of FIG. 27C onto wires “D0-3”. Although FIGS. 27A–27C show a 4-bit-wide variable, the variable may have any desired width, such as a 16- or 32-bit width.

According to one modification, ANSI C pointer and structure operations for a read operation may be directly implemented in a gate-level hardware description by configuring a series of flip-flops in hardware corresponding to the number of bits of memory address and another series of flip-flops corresponding in number to the number of bits of data to be read, and by providing control logic that enables data to be read from addressable external or on-chip hardware memory to output wires. ANSI C pointer and structure operations for a write operation may also be directly implemented in a gate-level hardware description by configuring a series of flip-flops in hardware corresponding to the number of bits of memory address and another series of flip-flops corresponding in number to the number of bits of data to be written, and by providing control logic that enables data to be written from addressable external or on-chip hardware memory to input wires. For example, a generalized circuit configuration may be defined in a library, and the number m of bits of data and the number n of bits of address would be directly derived from the ANSI C pointer to define the schematic for the circuit. Therefore, ANSI C pointer and structure operations may be automatically converted by a parameterizable library look-up using values for m and n to gate-level hardware descriptions similar to the schematics shown in FIGS. 22C and 23C for pointer operations and in FIGS. 24C, 25C, 26C, and 27C for structure operations, respectively.

In summary, prior art computer aided hardware design tools generate so many unhandled exceptions and language restrictions when presented with typical high-level language programs (e.g., a C-type program) that it renders such tools virtually useless for practical hardware design. Therefore, persons using such tools are required to read manuals and attempt to create work-arounds for portions of the high-level programming language that are not implemented. In contrast, a salient feature of the system in accordance with the invention is that almost any ANSI C (or high-level programming language translatable to a C-type programming language) can be compiled, thereby obviating the need for the user to read manuals and attempt to create work-

US 6,226,776 B1

27

arounds for portions of the high-level programming language that are not implemented. Existing programs in a high-level programming language can typically be fed directly into the system in accordance with the invention without any error messages being generated.

Hardware generated using prior art computer aided hardware design tools is typically based on simple line-by-line, table-driven translation and as such does not contain the necessary widely scoped optimizations that produce practical and useful hardware designs for most applications. The system in accordance with the invention supports all functions of the high-level programming language in which the hardware design is rendered at compile time and re-uses HDL functional blocks for hardware design efficiency.

Design output of the prior art computer aided hardware design tool can be in so-called "behavioral HDL," which in many cases cannot be synthesized by existing synthesis programs into a gate-level representation of the hardware. The system in accordance with the invention generates register transfer level HDL that can always be synthesized.

Prior art computer aided hardware design tools attempt to generate designs that only apply to a single manufacturer hardware family, such as the creation of an XNF file for the Xilinx XC4000 FPGA family. In contrast, the system of the invention converts to a synthesizable HDL which can be synthesized by standard synthesis programs for implementation on hardware platforms of any one of a number of manufacturers, including Altera, Xilinx, Actel, Texas Instruments, LSI Logic, VLSI Technology, Lucent Technologies, NEC, Hitachi, Toshiba, Fujitsu, and others.

Finally, the language employed by prior art computer aided hardware design tools has added and limited so many constructs that it is difficult to consider the employed language as the high-level language it purports to be. That is, the resultant language is more like a new specialized HDL. In contrast, the system in accordance with the invention enables the user to truly use a highlevel programming language, such as ANSI C, to design hardware.

The table below summarizes which prior art attempts described earlier have which shortcomings and shows the advantageous features of the system in accordance with the invention. The table clearly evidences that the system of the invention is a unique contribution to the state-of-the-art.

28

ating system platform, such as a UNIX operating system platform. These contemplated alternative embodiments and other modifications, changes, and adaptations are intended to be comprehended within the meaning and range of equivalents of the appended claims.

What is claimed is:

1. A method for converting a C-type language program to a hardware design, comprising the steps of:
creating an algorithmic representation in a given C-type programming language corresponding to a preliminary hardware design;
compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:
compiling a C-type program control flow into an HDL state machine; and
assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design;
configuring in the HDL synthesizable design an interface for a gate-level hardware representation;
determining the presence of any C-type pointers in the C-type program; and
compiling any C-type pointers and pointer indirection into an HDL state-machine-based memory access protocol.
2. The method of claim 1, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.
3. The method of claim 2, further comprising the step of using physical design tools to implement the gate-level representation as an actual hardware implementation.
4. The method of claim 1 wherein the C-type programming language is selected from among the group of C-type programming languages consisting of ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C.
5. The method of claim 1 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.
6. A method for converting a C-type language program to a hardware design, comprising the steps of:

Feature	Invention	TMCC	JRS	NLC	Handel-C	XC
1. Substantially full ANSI C	Yes	No	No	No	No	No
2. Logic design optimized	Yes	No	No	No	Yes	Yes
3. Design synthesizable	Yes	Yes	No	Yes	Yes	Yes
4. Targets general HDL	Yes	No	Yes	No	No	No
5. Is high-level programming language (not specialized new HDL)	Yes	Yes	Yes	Yes	No	No

It will be understood and appreciated that the embodiments of the present invention described above are susceptible to various modifications, changes, and adaptations. For example, the computer aided hardware design tool in accordance with the invention can also convert other high-level programming languages than those that are specifically identified to a synthesizable hardware description language. Also, although the preferred embodiments have been described for a personal computer (PC) operating system platform, it will be appreciated that the computer aided design tool in accordance with the invention can alternatively be adapted for execution on any other type of oper-

creating an algorithmic representation in a given C-type programming language corresponding to a preliminary hardware design;
compiling the C-type programming lanauage preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:
compiling a C-type program control flow into an HDL state machine; and
assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design;

US 6,226,776 B1

29

configuring in the HDL synthesizable design an interface for a gate-level hardware representation; determining the presence of any non-addressable variables in the C-type program; and mapping any non-addressable variables onto hardware registers formed in HDL.

7. The method of claim 6 wherein the C-type programming language variables include at least one of integer, floating point, pointer, enum, struct, and union variables.

8. The method of claim 6, further comprising the step of 10 synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

9. The method of claim 8, further comprising the step of 15 using physical design tools to implement the gate-level representation as an actual hardware implementation.

10. The method of claim 6 wherein the C-type programming language is selected from among the group of C-type programming languages consisting of ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C.

11. The method of claim 6 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

12. A method for converting a C-type language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given C-type programming language corresponding to a preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design; configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any addressable variables in 40 the C-type program; and

mapping any addressable variables onto at least one of addressable hardware memory and an addressable array of hardware registers formed in HDL.

13. The method of claim 12 wherein the C-type programming language variables include at least one of integer, floating point, pointer, enum, struct, and union variables.

14. The method of claim 12, further comprising the step of 45 synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

15. The method of claim 14, further comprising the step of using physical design tools to implement the gate-level representation as an actual hardware implementation.

16. The method of claim 12 wherein the C-type programming language is selected from among the group of C-type programming languages consisting of ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C.

17. The method of claim 12 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

18. A method for converting a C-type language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given C-type programming language corresponding to a preliminary hardware design;

30

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design; configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any complicated C-type mathematical operations in the C-type program; and compiling multiple occurrences of any complicated

C-type mathematical operation into a given HDL functional block that implements such an operation invoked on each occurrence by different states driven by the state machine.

19. The method of claim 18, further comprising the step of 20 synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

20. The method of claim 19, further comprising the step of using physical design tools to implement the gate-level representation as an actual hardware implementation.

21. The method of claim 18 wherein the C-type programming language is selected from among the group of C-type programming languages consisting of ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C.

22. The method of claim 18 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

23. A method for converting a C-type language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given C-type programming language corresponding to a preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design; configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any recursion in the C-type program, and wherein at least one of an identifiable direct and indirect recursive function call is compiled into an HDL state machine which has an interface to a stack implemented utilizing one of external and internal memory and an array of hardware registers, wherein the state machine has a stack pointer implemented utilizing a hardware register employed to store and restore local variables of the recursive function and other recursive function information in the stack.

24. The method of claim 23, further comprising the step of 60 synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

25. The method of claim 24, further comprising the step of using physical design tools to implement the gate-level representation as an actual hardware implementation.

26. The method of claim 23 wherein the C-type programming language is selected from among the group of C-type

US 6,226,776 B1

31

programming languages consisting of ANSI C, B, C++, Java, Kernighan & Ritchie C, and Objective C.

27. The method of claim 23 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. 5

28. A method for converting a high-level language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given high-level programming language corresponding to a preliminary hardware design;

translating the high-level programming language preliminary hardware design into a C-type programming language preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design; configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any C-type pointers in the C-type program; and

compiling any C-type pointers and pointer indirection into an HDL state-machine-based memory access protocol.

29. The method of claim 28, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

30. The method of claim 29, further comprising the step of using physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

31. The method of claim 28 wherein the high-level programming language is selected from among the group of C-type programming languages consisting of APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC. 40

32. The method of claim 28 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. 50

33. A method for converting a high-level language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given high-level programming language corresponding to a preliminary hardware design;

translating the high-level programming language preliminary hardware design into a C-type programming language preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design;

32

configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any non-addressable variables in the C-type program; and

mapping any non-addressable variables onto hardware registers formed in HDL.

34. The method of claim 33 wherein the C-type programming language variables include at least one of integer, floating point, pointer, enum, struct, and union variables.

35. The method of claim 33, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

36. The method of claim 35, further comprising the step of using physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

37. The method of claim 33 wherein the high-level programming language is selected from among the group of C-type programming languages consisting of APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC. 20

38. The method of claim 33 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM. 30

39. A method for converting a high-level language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given high-level programming language corresponding to a preliminary hardware design;

translating the high-level programming language preliminary hardware design into a C-type programming language preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design; configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any addressable variables in the C-type program; and

mapping any addressable variables onto at least one of addressable hardware memory and an addressable array of hardware registers formed in HDL.

40. The method of claim 39 wherein the C-type programming language variables include at least one of integer, floating point, pointer, enum, struct, and union variables.

41. The method of claim 39, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

42. The method of claim 41, further comprising the step of using physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

43. The method of claim 39, wherein the high-level programming language is selected from among the group of

US 6,226,776 B1

33

C-type programming languages consisting of APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

44. The method of claim 39 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

45. A method for converting a high-level language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given high-level programming language corresponding to a preliminary hardware design;

translating the high-level programming language preliminary hardware design into a C-type programming language preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design;

configuring in the HDL synthesizable design an interface for a gate-level hardware representation;

determining the presence of any complicated C-type mathematical operations in the C-type program; and

compiling multiple occurrences of any complicated C-type mathematical operation into a given HDL functional block that implements such an operation invoked on each occurrence by different states driven by the state machine.

46. The method of claim 45, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

47. The method of claim 46, further comprising the step of using physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

48. The method of claim 45 wherein the high-level programming language is selected from among the group of C-type programming languages consisting of APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

49. The method of claim 45 wherein the hardware description language is selected from among the group of

34

hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

50. A method for converting a high-level language program to a hardware design, comprising the steps of:

creating an algorithmic representation in a given high-level programming language corresponding to a preliminary hardware design;

translating the high-level programming language preliminary hardware design into a C-type programming language preliminary hardware design;

compiling the C-type programming language preliminary hardware design into a hardware description language (HDL) synthesizable design, wherein the step of compiling comprises the steps of:

compiling a C-type program control flow into an HDL state machine; and

assigning input/output as defined in the C-type program to specific wires in the HDL synthesizable design;

configuring in the HDL synthesizable design an interface for a gate-level hardware representation; and

determining the presence of any recursion in the C-type program, and wherein at least one of an identifiable direct and indirect recursive function call is compiled into an HDL state machine which has an interface to a stack implemented utilizing one of external and internal memory and an array of hardware registers, wherein the state machine has a stack pointer implemented utilizing a hardware register employed to store and restore local variables of the recursive function and other recursive function information in the stack.

51. The method of claim 50, further comprising the step of synthesizing the HDL design into the gate-level hardware representation using a synthesis program to interpret the HDL design.

52. The method of claim 51, further comprising the step of using physical design tools to implement the gate-level hardware representation as an actual hardware implementation.

53. The method of claim 50 wherein the high-level programming language is selected from among the group of C-type programming languages consisting of APL, Ada, Algol, B, Basic, Kernighan & Ritchie C, C++, CLOS, COBOL, Clu, Common Lisp, Coral, Dylan, Eiffel, Emacs Lisp, Forth, Fortran, IDL, Icon, Java, Jovial, Lisp, LOGO, ML, Modula, Oberon, Objective C, PL/I, PL/M, Pascal, Postscript, Prolog, Python, RTL, Rexx, SETL, Simula, Sather, Scheme, Smalltalk, Standard ML, TCL, and TRAC.

54. The method of claim 50 wherein the hardware description language is selected from among the group of hardware description languages consisting of Verilog, VHDL, ABEL, CUPL, AHDL, MACHX, and PALASM.

* * * * *